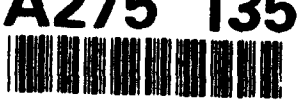




# EMPIRICAL STUDIES OF THE VALUE OF ALGORITHM ANIMATION IN ALGORITHM UNDERSTANDING



A THESIS
Presented to
The Academic Faculty

by

Andrea Williams Lawrence

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in Computer Science



Georgia Institute of Technology
August, 1993



# EMPIRICAL STUDIES OF THE VALUE OF ALGORITHM ANIMATION IN ALGORITHM UNDERSTANDING

Approved:

_____
Albert Badre, Chair

_____
James Foley

_____
Larry Hodges

_____
John Stasko

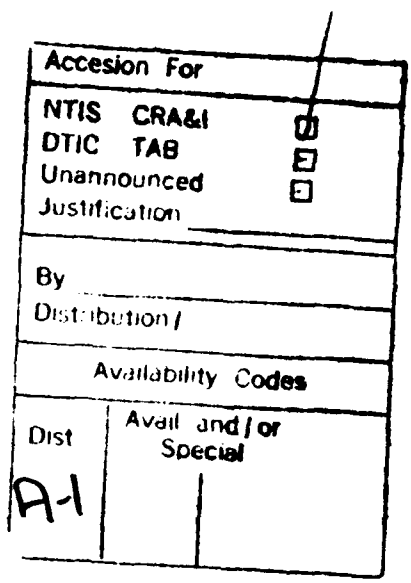
_____
Neff Walker

Date approved by Chair _____

# DEDICATION

— To my parents, who taught me the importance of education before I ever entered school.





# ACKNOWLEDGMENTS

I acknowledge the support and assistance of my family, friends, and the faculty of Georgia Tech. Pete Jensen and Ed Rumiano encouraged me to enter the program. Janet Kolodner and Dean Peter Freeman encouraged me to remain.

The members of my committee, Dr. James Foley, Dr. Larry, Hodges, Dr. John Stasko, and Dr. Neff Walker have been invaluable. Dr. Albert Badre, chairman of my committee, has guided and advised me at all times. The road to the dissertation is a hard one which is made easier to travel by the assistance of others.

I especially acknowledge the support of my reunion group, Yvonne Bernal, Carolyn Fuller, Dorothy Clinkscales, Saundra Harris, and Michelle Barnes, who prayed me through to this point. Yvonne deserves a special mention since she has served as "other mother" to my daughters. My daughters, Deirdre, Allegra, and Valerie, had constant faith in my ability to succeed. My office mate, Jeanette Allen was never too busy to discuss a new idea or help solve a problem,

Teresa Edwards and Benjamin Martin of Spelman College were unfailing sources of support, as was my brother-in-law, Johnny Houston who advised and encouraged me in the pursuit of the Ph.D.

Many of my fellow students made me welcome and eased my transition into the Georgia Tech community. They include Mary Jane Willshire, Johne Parker, Byron Jeff, and Vernard Martin.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

We present a series of studies on using algorithm animation to teach computer algorithms. These studies are organized into three components: eliciting students' preferences, developing algorithm animation guidelines, and evaluating the effects of using algorithm animation in the classroom. Many systems for creating computer animations have been designed. These systems reflect the designers' confidence that visual representation is a valuable technique for conveying conceptual knowledge. However, little formal experimentation has been carried out to determine whether such animations are beneficial in teaching the algorithms presented. In addition, formal guidelines have not been developed for either design or use of these animations. This work addresses both concerns. We found that student preference for animation does not predict student performance on a post-test to measure learning. We examined several types of labeling (data, steps, actions) and found a variety of effects. One of these effects was that a text description of the algorithm increased accuracy on conceptual questions. Also, students are more accurate on a post-test if they are allowed to design their own data sets than if they use experimenter-defined data sets. Finally, use of the animations was evaluated in a classroom-type setting. All students were given a lecture on an algorithm. Adding a laboratory in which the subject controlled the animation and data sets led to more accurate performance than a lecture only or than a lecture with experimenter-preprepared data sets. This work has implications for the design and use of animated algorithms in teaching computer algorithms and for the design of laboratory experiences for beginning computer science courses.

# CHAPTER I

# INTRODUCTION

Modern computer techniques allow the creation of animations of programs and algorithms. These animations provide a moving pictorial representation of the progress of the solution to some problem. Many program or algorithm animators are convinced of the truth of that old Chinese proverb, "A picture is worth 10,000 words." Psychological studies of learning, memory, and problem solving also support this folk wisdom. However, folk wisdom provides scant direction for using the new technical advances in visual presentation. Animation is now a feasible tool for teachers. However, minimal empirical evidence supports using animation in teaching. In particular, few studies have been done which demonstrate support of animation for teaching algorithms and computer programming.

Research in this area continues. Various aspects of visual programming are being investigated. Programming classes in some colleges and universities have adopted program or algorithm animation. Although both students and faculty report satisfaction with the use of these aids, this is a subjective measure rather than an objective one. These studies have not proven that animation yields better learning or understanding.

Before any instructional technique is advocated, much less adopted, it must be proven to provide educational advantage. Assertions such as "students liked the system," or that "the system clearly provided faster learning" are not sufficient. New techniques or equipment are all too readily accepted as a panacea without proof of effectiveness.

Even before testing the animation in the classroom, considerable thought must go into the design of the animation.

Guidelines exist for the design of interfaces and for the design of graphical presentations. However, the design of an animation must account for the additional complexity of change over time. This dimension places an extra burden on the watcher's memory, to

associate the current state of the animation with previous states. As a result, animations have been based on intuition and which data representations are available in the system.

This research addresses the use of animated algorithms in teaching computer algorithms. Three aspects of this use are considered.

- The first is to elicit student preferences about animations.
- The second is to validate design guidelines for animations.
- The third is to confirm the value of these animations in teaching.

Little has been done to determine which types of animations are most effective.

Student preferences were elicited on data representations, labeling schemes, etc. Other experiments completed in this research examined this issue of developing effective animations. The variables included data representation, data set size, color, data element labels, text labels of algorithmic steps, and control mode.

Once the animation has been developed, the research must be concerned with how the animations should be used in teaching these algorithms. An experiment of various presentation styles for animations in teaching was conducted. There are several possible approaches, ranging from classroom lecture examples through supervised laboratory presentations to unsupervised or discretionary use. This research compares some of these options.

## 1.1 The Problem

Many researchers believe that animations should help in understanding the algorithm through visual presentation of the algorithm in action. In spite of the researchers' intuition, the actual value of these animations in learning algorithms has not been demonstrated. Many studies either perform informal analysis or show little or no effect.

Three reasons for the scarcity of validating studies are the efforts necessary to design good animations, implement the design, and use the animation appropriately in the classroom. The second concern, implementing the design, has been the focus of much recent

research. The last several years have seen the emergence of systems which allow the rapid development of animations. These systems vary in level of difficulty and amount of programmer involvement. Many such systems have been presented at national meetings and conferences.

With the development of these systems and this new technology, the other two issues, good design and appropriate use, may now be addressed. Unlike many areas of the human computer interface, formal design guidelines do not exist for animations. These two issues include broad classes of knowledge such as procedural and operational learning as well as how mappings from animation to algorithm are performed. The complexity of the issue has discouraged researchers from formal experimentation.

These two issues are examined in three stages. The first two address design–the first from the perspective of user preferences, the second from the perspective of learning performance. The third stage examines appropriate use of animations in teaching. The variables controlled were representation of data, data set size, color, the optimal combination of textual and visual cues, control, and presentation style in the classroom.

## 1.2 Contributions

The studies presented here answer many questions which have been raised about animations, but have not been previously addressed. We determine student preferences and the relationship of preference to performance. We address design issues and the implications of the nature of the task animated. In so doing, we develop guidelines for the design of animations. We also address the issue of how best to use such animations for teaching algorithms. Suggestions are made for the use of animations on teaching.

Studies such as these provide objective measures of the value of the use of these animations rather than subjective opinions. If animations are to become part of the teaching process for algorithms, guidelines such as the ones presented here should be used to maximize effectiveness. In addition, the use of these animations should be done in the manner which will optimize performance. Thus animations can be used to the best effect when they are

added to the presentation of computer algorithms.

## 1.3 Overview

In Chapter 2 we present a review of relevant literature from the field. Topics covered include advantages of pictures over text, graphical perception, labeling, visual techniques in concept modeling, incidental learning and animation systems.

A series of experiments was run to investigate various issues with regard to algorithm animation. These experiments used many of the same materials and methods. These are collected into Chapter 3. The chapter also includes a description of the XTango animation environment which was used to create the animations used in the experiments. Also described is the on-line test software developed for the series of experiments.

The experiments are divided into three stages. The first stage examined student preferences by showing students several videotapes and animations and then eliciting their preferences. Chapter 4 describes the first stage of the experimental sequence, preference studies based on three representations of data in sorting algorithms. The representations used were vertical bars, horizontal bars, and dots. In these studies students were asked to compare and rate the various representations. The result of these studies was that students preferred vertical bars. A formal experiment conducted to measure the effect of these different data representations and data set sizes indicated that there was no significant effect on performance based on representation, but that the medium data set size produced the best mean results on the post-tests.

The second stage presents exploration of the design and task elements of the research. Chapters 5, 6, and 7 describe experiments dealing with the design of animations. In Chapter 5, we present the results of an experiment based on the labeling of data elements. This did not prove to be a significant factor in the design of the animations. In Chapter 6 we discuss an experiment which involved contrast of passive (observational only) use of animations with active (user provided data). Subjects who generated examples performed at a higher level in the learning of Kruskal's Minimum Spanning Tree Algorithm. Interactive

use of animations proved to be a significant factor in understanding and application of the algorithms. Chapter 7 examined algorithmic steps and color labeling as design issues. Results indicate that monochrome displays and textual steps accompanying the algorithm aid understanding.

The third set of experiments focused on issues of how to use the animations in teaching algorithms. A series of small studies appear in Chapter 8. They investigated questions of how text should be used with the animation. In this experiment, text was a written description of the algorithm. One study concerned the order of presentation of text and animation. Text followed by animation led to the best results. Other studies were exploratory in nature. They involved discovery learning using the animation alone, text alone, and text plus animation. Animations alone were found to be adequate to present simple algorithms, but insufficient for more complex algorithms.

The guidelines and results from all the previous experiments were used to develop animations for the final experiment. The final experiment, presented in Chapter 9, focused on using the animations in the teaching process. Animations were used as examples to accompany lectures, in passive laboratory sessions, and in active laboratory sessions. Use in active laboratory sessions led to higher scores on post-tests.

Chapter 10 summarizes the results of these experiments. Guidelines are provided for the design of animations. Suggestions are given for appropriate use of animations in classrooms. This series of empirical studies indicates that animations can be designed to produce maximal impact and that optimal use of these animations employs student control of and interaction with the animation.

# CHAPTER II

# REVIEW OF THE LITERATURE

## 2.1 Background

We conducted a series of experiments to elicit student preferences for animations, to develop design parameters for animations, and to validate the use of animations in the classroom. Before describing these experiments, we examine research on the use of visuals in teaching and learning. This includes exploration of reasons supporting the use of pictures, how visual techniques increase learning and understanding by aiding the construction of concrete models, and the types of algorithm visualization available. We survey empirical evidence related to the area. We discuss work in the area of instructional graphics. Finally we discuss uncharted areas in the domain of algorithm animation.

## 2.2 Pictures versus Text

Few studies have been done to compare animation, which is the dynamic presentation of visual information, to text. However, static presentation of visual information (such as pictures and graphs) have well documented advantages over text. The following list is only an abbreviated selection of facts and studies which demonstrate the superiority of pictures over text for basic perceptual and cognitive tasks.

### 2.2.1 Advantages of Pictures

1. The content of a picture can be understood much more rapidly than the content of a text segment [29].

2. The human visual system is very powerful. It processes images with both speed and accuracy. Graphical techniques allow the involvement of this system in the learning

process [10].

3. Concept learning may take place without the use of text. This is illustrated by concept formation in prelinguistic children described by Piaget as cited by Baggett [7].

4. Pictures are remembered better than words. This result lays a foundation for the use of graphical techniques in teaching.

5. Mayer [37] suggests that students learn more from text coupled with illustrations than from text without illustrations. Students learning high school physics concepts recalled much more of the conceptual information and solved more of the transfer problems when provided with a diagram model.

6. The graphical approach may encourage attention to aspects of a program that otherwise might go unnoticed [12].

7. Data indicates that animated visuals may enhance initial encoding and subsequent retrieval [49].

8. Informal experiments have been done to determine if graphical representations increase insight and understanding. These experiments indicate that a visual representation definitely increases comprehension. Commercial systems such as spreadsheet programs indicate that visualization of the process leads to potential success and reduces user anxiety [6].

9. Myers [38] suggests that graphics should be used because graphics lend themselves to a higher-level description of the actions desired. This can be accompanied by less emphasis on syntax at a higher level of abstraction. Thus a graphical system may provide information which can aid in understanding of the current state. In addition, programming languages can be difficult to learn. Graphical systems allow the complex content to be demonstrated without requiring possession of all the skills of the language involved.

10. Larkin and Simon [31] described three cognitive processes which are used by problem solvers who use an external representation in the problem solution. They are search, recognition, and inference. Visual representations can carry much of the inferential knowledge.

11. Cunniff and Taylor [20] conducted studies involving program comprehension by novice programmers. Their results with FPL (a graphical representation similar to flowcharts) indicated that speed was enhanced by the use of the graphical representation.

12. Allen [1] found that having a visual representation of a programming language increased programming accuracy in a visual programming environment.

### 2.2.2 Disadvantages of Pictures

Pictures alone are not sufficient for conveying all concepts. Baggett [7] stresses that the correct mix of visual and verbal information is essential for maximum effectiveness. Her studies indicate that visual material creates a more varied association net in memory of already learned concepts. She suggests that visual material be used to provide more information about concepts which already have been learned. Verbal information is used to assure that the correct connections are made with the visual presentation. This implies that the ideal animation may require text or verbal additions to aid in forming the concepts desired.

The question arises as to whether certain features of information may be best presented by graphics or by text. Feiner and McKeown [22] suggest that abstract information and rationale should be represented by text, while actions and location as well as physical attributes should be represented by graphics.

Two possible types of information that could be included in the animation are the steps in the algorithm and the state of the data structure. Applying Feiner's suggestion to the design of animated computer algorithms would suggest that conceptual steps of an algorithm should be text labeled while actions taking place on the data structures would be represented in a graphical manner.

Another problem with pictorial learning aids is that they may lead to misconceptions.

This is demonstrated with transfer tasks. Transfer tasks are often used to test the ability of the learner to extract basic principles from the task or examples used in learning. A frequently cited example [24] of this type of task is the problem on the General who must attack a city that may be entered by several roads, but is unable to send a large force along any road. This problem is transferred to a medical dilemma where a X-ray dosage large enough to destroy a tumor will also kill the patient if delivered to one spot.

Ross [52] stressed that similarities between the original problems and the transfer tasks be based on structural and not superficial similarities.

Another challenge for designing animations is that incidental learning may be increased. In contrast most teaching is aimed at intentional learning (learning of a presented concept or process). Unfortunately graphic aids can unintentionally reinforce superficial similarities. Rieber [46] observed that students were able to acquire understanding of incidental information without decreasing intentional learning. His experiments demonstrated that incidental learning was higher with animation than with static graphics. Unfortunately, this incidental learning also led to misconceptions, for example they developed incorrect extensions of the principle of the effects of gravity when viewing a gravity-free animation. In general incidental learning may be detrimental, in that it may interfere with intentional learning. On the other hand, it may be helpful in that general principles are learned which may be applied to several related problem situations.

## 2.3 Design of Pictures

To obtain the maximum benefit of graphics at the minimum cost requires pictorial design based on principles of graphical perception. Graphical perception may be defined as the "visual decoding of information encoded in graphs" Choosing the appropriate encoding of the information can be difficult. Recently, Tufte [62] has been a leader in pointing out many bad practices in graphing. Cleveland and McGill [19] have been active in practical experimentation in the area of graphical perception.

The design of graphs and other non-textual aids in presenting data is a complex area

which has received considerable attention in recent years. It is generally believed that the format of the data affects the ease and accuracy of decoding the information which the graph represents. Although this debate has raged for over 100 years, recently there has been an upsurge in formal experiments on the presentation of scientific data. Practitioners of the field have often relied upon intuition and experience to determine the best form for presenting a particular collection of data.

The domain of graphing scientific data includes research on the efficacy of a certain design style for particular data. Mackinlay [33] points out that a graphical presentation must meet both expressiveness (expressing the desired information) criteria and effectiveness criteria (which representation is best at maximizing the capabilities of the computer system and the human visual system). Graphical presentations use graphical marks, including points, lines, and areas, to encode the desired information.

Preece [43] discusses some issues in the domain of displaying quantitative information graphically. She points out that how the data will be interpreted by the user is more important than semantic correctness.

Through experiments, Cleveland and McGill [19] have developed a hierarchy of the elementary perceptual tasks in graphical perception, based upon accuracy in experimental situations.

The hierarchy of discrimination tasks from highest accuracy to lowest accuracy is:

1. Position along a common scale
2. Position along nonaligned scales
3. Length
4. Direction, angle
5. Area
6. Volume, curvature
7. Shading, color saturation

So for instance, they found that the human eye is able to differentiate position more easily than length and length more easily than direction and angle. Application of these results can be used to help determine the representation of data for a particular task. Using the most accurately judged form will increase the chances of a correct perception. For example, bar charts are preferred to pie charts since this replaces angle judgments by position judgements.

In addition to selecting the best method to be used in representing data, identifications may be aided by texture, color, labels, and other cues such as position in the scene. However, experiments have shown that the primary recognition is line based [9]. That is, the lines which make up a figure or graph are the key to deciphering its meaning.

Wickens and Andre [66] point out that color can aid grouping of objects, but slows down focused attention tasks. Integration performance is also slower with colored objects. When an object is to be separated from the group, color cues aid attention.

The application of these findings to algorithm animation seems to indicate that data judgments would be best made based on position along a common scale or identical but non-aligned scales or length. Representing data size as a pie segment or sloped line or an area would be second choice. Color contrast could be useful in singling out some object to be noticed in the process of the algorithm, but should not be overused because this would slow down the focused attention.

Sparrow [54] points out that the form of the presentation can influence the recall of information. Studies done in designing icons indicate that the type of representation is a significant factor in the ease of understanding the concept represented [32]. Although common practices in style and method of representing data have arisen, neither a body of formal experimentation, nor the subsequent formal guidelines have been created. Such formal experiments must be done before any particular representation can be adopted and accepted.

### 2.3.1 Influence of Labeling

A recurrent theme in the problem solving literature is that the more salient is the information derived from the example, the better are the problem solving results. An obvious method for increasing salience is to label or highlight important features.

The influence of labeling in problem solving has a long history. One often cited example is Duncker's traditional candle problem in which a box of thumbtacks and a candle are given to a problem solver. The problem is to attach the candle to the wall in such a position as to allow it to burn. This problem is more easily solved when the box of thumbtacks is labeled "box." Students given this problem with the labels are more likely to discover that the box can be thumb-tacked to the wall and used as a base upon which to set the candle [25].

Catrambone and Holyoak [18] note that emphasizing structural features of a problem and labeling subgoals aid the transfer of the problem solving technique to new problems. Ross also [51] emphasizes the necessity of distinguishing the critical features in some manner in order to extract the right information from examples for later problem solving. Much of the psychology based literature refers to labeling as the inclusion of textual names or descriptions, but Maguire [34] summarizes a body of research which stresses other types of cues such as highlighting, shape coding, color coding, and blinking.

### 2.3.2 Use of Visual Techniques in Concept Modeling

Modeling new concepts can increase learning and understanding. At the higher, conceptual level, visual information can aid in learning and understanding abstract concepts. It has been demonstrated that using concrete models was helpful in learning technical or unfamiliar material and aided transfer to new situations. Visual models used in LOGO and BASIC computer models supported this result especially for transfer problems. Visual models in the area of physics have also been found to increase concept acquisition [37].

Visual techniques aid in the construction of concrete models. Mayer [37] suggests that key factors in measuring the effectiveness of model usage are learner characteristics,

material to be learned, instructional method and learning outcome performance. Models must be coherent, concrete, conceptual, correct and considerate to be effective [36].

### 2.3.3 Instructional Graphics

Many studies have been done in the area of illustrated text. Weidermann [64] reminds us that Levie and Lentz reviewed forty-eight experimental studies with the results being overwhelming in favor of illustrated text. Thus pictures with text can help to facilitate learning and retention. This result has been demonstrated for varied tasks and learners. He cites Duchastel's 1978 [21] statement of the three roles of pictures being: (1) Pictures can attract the learner's attention; (2) Pictures can help the reader to understand information that is hard to describe in verbal terms; and (3) Pictures can reduce the likelihood that acquired information is forgotten, perhaps as a consequence of an additional encoding in pictorial memory.

Designers of instructional graphics, such as charts, diagrams, and graphs, must consider the method of instruction, expected outcomes, and the conditions under which instruction is to take place. Winn [68] reminds us that different graphical forms convey different meanings. Two basic functions of graphics are to simplify the complex, and to make the abstract more concrete. The *best* design carries out these functions. An important component of the best design is the correct representation of the data. Winn stresses that representation effects the meaning of the graphic. Such representation may be through verbal labels, drawings, or symbols. Some graphics have conventions, such as those in graphs which arise from mathematical usage setting larger values at the top for the Y values and on the right hand side for the X values. Winn also notes that mental models of conceptual domains in science can be developed by using graphics[67]. Such strategies may be taken into account when designing animations in order to help build concrete mental concepts. For example, it would be counter-intuitive to design an animation in which values of the array grew in value from right to left on a line which resembled a graphical X axis. Another example is that arrows can be used to focus attention. The concept to be conveyed must effect how

the animation is constructed and presented.

A computerized instructional application must be well-designed to succeed [44]. Rambally and Rambally stress the need for more guidelines to guide the design of the computer interface for computer assisted instruction. Rambally and Rambally provide some general guidelines for design of such computer assisted instructional materials. Among these are clearly organized easy to understand screen designs, material of an appropriate level, highlighting of important information, avoiding clutter, and effective use of color, shape and size codings. In designing the XTango animations, these guidelines were considered. The concepts of visual coding and highlighting were used in these designs. These guidelines will help in the preparation of software to be used in instruction, but evaluation is also essential.

Palmiter [41] gives empirical evidence that animated demonstrations are best combined with non-redundant text. She cites studies in which such combinations have proven more effective than animated demonstrations alone or text alone for immediate and delayed tasks. Novice users find such demonstrations more effective than the long term or expert user. Thus, the animations should be created with the eventual user in mind. Such animations are best, she states, for graphical tasks which are not highly complex. Palmiter also reports user preference for animation demonstrations over text.

## 2.4 Visualization

Animation may be formally defined as "the rapid sequential display of pictures or images, with the pictures changing gradually over time" [60]. Animation therefore presents an extra dimension for design beyond that of static graphics. As a result, previous research in graphic design has limited applicability. The previous research gives very general guidelines for design. It also suggests some directions in which animation could successfully be employed. These results must be carefully interpreted in view of the unique opportunities animation provides.

One important decision must be the content of the animation. Several researchers have attempted to characterize and analyze the types of visualization which can be pre-

sented. Myers [38] presents the categories of visual programming, programming by example, and program visualization. On the other hand, Stasko and Patterson [60] present the categories of data structure display, program state visualization, program animation, algorithm visualization, and algorithm animation. Baecker [6] defines program visualization as "the use of the technology of interactive graphics and the crafts of graphic design, typography, animation and cinematography to enhance the presentation and understanding of computer programs. Program visualization is related to but distinct from the discipline of visual programming which is the use of various two-dimensional or diagrammatic notations in the programming process" [12]. This research deals with the area of algorithm animation, which may be described as a form of program visualization.

### 2.4.1 Systems using Animation

Several systems have been developed which employ animation techniques in teaching computer science. These systems vary in the degree of control given to the user and to the designer of the animation.

- Often cited is Baecker and Sherman's [5] film "Sorting Out Sorting", shown at SIGGRAPH '81, which is an early example of algorithm visualization.

- Bocker, Fischer and Nieper [10] developed a system (KAESTLE) which is intended to aid understanding of LISP through visualization of data structures and the dynamic behavior of a running program.

- Brown and Sedgewick [17, 16, 12] developed the Balsa system for algorithm analysis. Brown University used the system in computer science classes including introductory programming, data structures, and graphics. The successor system, Balsa-II, allows the user to adapt his/her view of the animation. Users may execute in single step, adjust window size, zoom in and out, and modify the data representation. For example, in a sort the elements may be shown as "sticks" or as "dots." The authors contend that a single view may not be best for all algorithms, but also believe that a view should be flexible enough to be used for several algorithms. The environment allows

an investigation of the "dynamic behavior of programs." It is essential that updates be done rapidly.

- Marc Brown [13, 14, 15] developed the Zeus algorithm animation system which allows the user to control the data, the type of views, and algorithm execution. It allows for "interesting events" to be identified and supplied with parameters. This system incorporates objects, strong-typing, parallelism, and graphical development of views. It also allows the use of color and sound in the animation of parallel algorithms.

- Stasko's Tango system [56, 58] allows the ready development of animated visualizations of computer programs. Because computer programs may be difficult to interpret in their usual textual format, an animated graphical presentation of the program may aid interpretation of the program's purpose and meaning. Animated views aid in understanding programs, evaluating existing programs, and developing new programs. The Tango system allows animation of various programs without requiring a different animation routine for each program. Programs in sorting, searching, graph and tree manipulations, and classic problems such as the Towers of Hanoi and producer-consumer ring buffer have been animated.

- Vanneste and Olivie [63] developed an intelligent environment for learning Pascal programming. This system allows the student to visualize the process of execution, and in so doing observe the effect of the algorithm.

- The ALADDIN (ALgorithm Animation Design and Description using InteractioN) system [27] allows the creation of animations of algorithms. The graphics are designed interactively with a graphical editor. An arbitrary algorithm written in Modula-2 can be demonstrated as an animation without the user having to program the graphics or other parts of the animation. The user controls the appearance of the algorithm. Objects are specified graphically by the user.

A related area is that of data structure presentation or animation. Graphical displays of data structures were presented as early as 1980 by Myers in the INCENSE system [40].

Baskerville's GDBX [8] extends this idea. While stressing the importance of data structures as fundamental to designing or understanding a program, GBDX integrates the display of data structures into a debugger. It also allows user control over the data structures presented, including such things as scrolling an array. Individualization of use is possible, providing a customized user environment.

### 2.4.2 Empirical Research

Although developers of algorithm systems are convinced of the value of animation systems, little actual experimentation has been done to measure the efficacy of teaching computer algorithms and concepts using an algorithm animation system. Studies have been carried out in some related areas such as solving algebra problems, programming, teaching computer-based tasks. teaching science, and teaching computer science. Results have been mixed.

Early studies using animated displays in solving algebra problems by Reed [45] indicated that an external lesson strategy was necessary. It was not sufficient to simply show the animated display. Combining the animation with an external lesson strategy focused attention on the pertinent features of the animated display.

Use of animation has not always proved instructionally effective. Rieber [49] study used animated visuals embedded in a lesson about Newton's Laws of Motion. No effect was shown for animation. A later related study [48] did show an advantage for the animation group over static and no graphic groups. However, this was linked to practice rather than to the type of visual aid. Rieber, Boyce and Assad [47] examined the effects of different levels of visual elaboration on adults learning a computer based science lesson. They found no significant differences.

An observational study by Badre, Beranek, Morris and Stasko [3, 4] indicated that students found algorithm animation valuable when used in conjunction with a computer science class.

Animations are also being used in the teaching of computer-based tasks. Palmiter

and Elkerton [42] compared animated demonstrations to written instructions. Results indicated that training took longer for written instructions, and scores on the immediate test were higher for the animated demonstration group. In contrast, the textual group performed faster in the delayed test. Correctness was better for the animation group during training, equivalent during the immediate test, and lower during the delayed test. One possible explanation for this is that the demonstration group used mimicking during training and the immediate test and did not encode the information as redundantly as the textual group.

In a second experiment, Palmiter and Elkerton [42] used animated demonstrations to help users learn about direct manipulation interfaces. Because users often fail to read written instructions or find them difficult, the experimenters felt that animated demonstrations would be helpful. These demonstrations skip the referential step needed to comprehend text while providing an immediate link between input and response. Palmiter and Elkerton taught subjects a variety of HyperCard authoring tasks using demonstration only, spoken text only, or demonstration and spoken text. Results indicated that the demonstration groups enjoyed their instructions more than the spoken text only group. During the training session, the demonstration groups were faster and more accurate; however, after a seven day delay the spoken text-only group was faster. Thus the best liked type of instructions did not yield the highest retention.

Stasko, Badre, and Lewis [57] present empirical study of the use of algorithm animations in teaching an algorithm involving the pairing heap data structure. Students in a text-only group were compared to students in a text-animation group. Although there was no significant difference in results on the post-test, the animation students felt the animation was a help in understanding the algorithm. Some also requested explanations to accompany the animation. The authors discuss the importance of providing a more active learning experience for the novice student, perhaps by allowing the student to construct an animation. They also suggest that optimal use of animations requires the student to understand both the algorithm and the mapping from the abstract computational algorithm

domain to the computer graphics animation domain used to present the algorithm.

Whitney and Urquhart [65] integrated computers into two courses at San Diego State University. In an algorithms course, normally a difficult course for students, MacBalsa (by Marc Brown) and AL (Algorithms Lab, Roger Whitney) were used. MacBalsa is an algorithm animation program. AL provided a timer, plotter, and least squares fitter to be used in the study of algorithms. Results were disappointing in that computer usage seemed to widen the gap between the strong students and the weaker students. This is attributed to the large cognitive load the computer placed on the weaker students.

Overall these studies show that animation used in *conjunction* with other aids or materials provides the learning and retention. Further studies are needed to determine when to use animations, the best format for the inclusion of animated algorithms in the curriculum, and the best representation for the data involved. A clear comparison of traditional teaching methods to teaching with the aid of algorithm animation is needed to lend support to the use of animated algorithms in teaching computer science. So far there is little evidence that animation helps learning. Before animations are developed for use in teaching algorithms, there must be statistical evidence of the value of this use.

## 2.5 Open Questions

Visualization alone is not enough. Presenting a visual representation of information does not guarantee that there will be successful human-computer communication. Semantic qualities may be omitted [35]. Brandenburg [11] states that the question of optimal graphics is NP complete, that is, calculating the best drawing is extremely expensive in a computation sense. There are many levels of abstraction and many possibilities to be considered. Research on the static presentation of data may offer some hints toward optimizing the presentation of animated algorithms [11].

Much progress has been made in providing systems and techniques which allow visualization of computer algorithms. Many researchers agree that the state of the art, while much advanced over the past few years, still has unanswered questions.

- Modifications of or access to data structures are not necessarily the same as algorithm operations. It is unclear which of these should be represented by the animation.
- Real-time performance may be difficult to provide due to hardware restrictions, complexity of the algorithm, and size of the data.
- Displays need detailed information about run time activities which may necessitate limitations in parameters and data sets [12].
- No one knows what a "nice layout" really is in this domain [6].
- This area is a new paradigm. Issues include the fact that it is difficult to specify what the data animation should look like and the question of at what points to update [39].
- Experts in the area note that design principles are needed. Agreement has not been reached on what is best in many aspects of animation visualization [6]. For graphs, Tufte [62] provided detailed design criteria and Mackinlay [33] automated the selection of representation and format. However, no equivalent work has been done for animation.
- Stasko [56] suggests that a need exists for a systematic approach to the use of animated algorithms.

In short, the best way to implement algorithm animation is unknown. Important issues are screen layout and representation of data, speed and amount of data to be used, and how often to provide update. Also important are questions of color, textual cues, accompanying textual explanations, and user control of the animation.

These questions of format and design are critical. Another key question is this: How much does the use of these techniques improve the learning and understanding of computer algorithms? Although both instructors and students have commented positively upon the use of animated algorithms, not everyone agrees that these techniques have a positive effect on performance. It would be comfortable to assume that algorithm animation is a positive influence upon the teaching of computer based algorithms. However, empirical

experimentation must be done to determine if animated algorithms are superior to standard teaching techniques.

In studying the use of these animations, it is also critical to investigate the user preferences, the effects of various representations and designs on performance, and the appropriate integration of animations into the classroom. All of these areas of concern were addressed in this research.

The first series of experiments dealt with student preferences. Two studies were carried out to elicit student suggestions and rankings of representations. These preferences concerned data representation, data labeling, text commentary, and other factors. Students preferences were compared to student performance in the next experiment. Unlike many areas of the human computer interface, formal guidelines do not exist to determine the best representation of data or the best data set size to use for maximum comprehension. Representation of a problem has long been recognized as extremely important to problem solution. Larkin and Simon [31] suggest that the key to solving a problem is to represent it correctly.

The scope of the problem as well as what features are to be stressed affect the best representation to use for graphical presentations of data. Position values are best expressed by dots, while length values may be better expressed by lines [33]. Work by Robertson [50] indicates that vertically stacked surfaces can show global correlation while horizontally aligned surfaces can show correlations in point values between individual values. Robertson [50] suggests a small data set size is best for introducing new concepts while a larger data set size is helpful for developing intuitive understanding of an algorithm's behavior.

The second series of experiments dealt with the questions of animated algorithm guidelines – they explored questions of speed, control, data set size, and representation. The results of these studies were used to prepare the version of the algorithm animation to test the advantages of incorporating the animated algorithms into the actual learning and comprehension of these algorithms.

The third area of concern was a formal comparison of the visual animation algorithm

systems and the "old-fashioned" or traditional way of teaching with teacher drawn sketches or or verbal descriptions. If animation is to be used, it must be integrated into the learning process. Various combinations of animation/no-animation and use of animations in lecture and/or laboratory can be studied. The third set of experiments dealt with this comparison.

# CHAPTER III

# MATERIALS AND METHODOLOGY

## 3.1 Introduction

The series of experiments presented in this research focus on various features of animations and on methods of using these animations in teaching computer algorithms. For simplicity of reference and to reduce redundancy, this chapter describes some of the methods and materials used. In particular, the subjects, basic experimental procedure, independent variables, dependant variables, and software are detailed.

## 3.2 Subjects

Subjects were student volunteers who received class credit for their participation in the experiment. Students from the Georgia State University were involved in the first two preference studies. Other subjects were students at the Georgia Institute of Technology. These students were engineering majors or computer science majors. They were selected because they represent a target audience for animations of this type since the classes they take require them to master many algorithms in the course of a quarter. Generally, if a mastery post-test was involved, the students had not been exposed to the algorithm in question beforehand.

## 3.3 Description of General Experiment

Experiments in this sequence generally involved individual sessions with students. These students were subjects in lower-division courses in Computer Science at the Georgia Institute of Technology. Students received class credit for participation in the experiments. In general, these sessions lasted for about one hour. The experimenter read a general descrip-

tion of the experiment. Subjects signed a consent form and filled out a brief demographic survey. A pretest was given to determine if prior knowledge affected scores.

Students were assigned randomly to cells in the experiment to prevent order affects. Generally, the student viewed animations of one or more algorithms. This was followed by a post-test designed to check understanding and application of the algorithm presented. During the early experiments students were asked for suggestions on improving the animations. These suggestions were used in selecting factors for later experiments.

The consent form appears in Appendix A. The preference survey used in the early studies appears in Appendix B.

Sample algorithm descriptions of Quick Sort and Selection Sort may be found in Appendix C and Appendix F. A sample of the pre-tests used is found in Appendix E, while samples of the post-tests used appear in Appendices D, G, I, J, K, and L.

### 3.3.1 Independent Variables

Variables which have been considered in this sequence are representation of data, data set size, color, labels of data elements and of algorithmic steps, interaction with the animation, and complexity of algorithms.

Experiments completed in this research involved several variables:

1. **Algorithms**

   The domains of sorting and graph algorithms were chosen for this sequence of experiments. Since there are many approaches to sorting and these approaches differ in complexity, this domain provides the opportunity to test the algorithm visualizations under varied conditions while maintaining a continuity of purpose. These algorithms differ in complexity with the majority requiring $O(n^2)$ or $O(n \log n)$.

   Because each algorithm has its own unique features, the animations must vary and the effectiveness of the animation in demonstrating the algorithm may be more for one than for another. To provide a different comparative approach a minimum spanning tree algorithm from the domain of graph problems [30] was also used. This issue is

examined in more detail in a separate section.

2. **Data Set Size**

   Data set size was tested since it was unclear what was the optimal size of data set sufficient to prove the point without outlasting student interest. Small, medium, and large data sets were used.

3. **Data Representation**

   The XTango system allows varied representations of data such as vertical bars, horizontal bars, and dots. These representations were examined to determine both students preference and which representation contributed to best performance.

4. **Color**

   Color can be used to focus attention and to aid grouping of objects. Studies have shown, however, that color may be a distractor in some situations and result in slower task completion. Color as an action cue was examined to determine whether it served as an aid or as a distractor in the XTango animations.

5. **Data Labels**

   Labels were investigated in two ways, as labels on the data itself and as textual labels which described the algorithmic steps. Labels on the data elements represent relative values.

6. **Algorithmic Text Labels** Textual labels were phrases which could be highlighted as that particular portion of the algorithm was carried out. They describe

7. **Interaction Mode**

   Interaction mode may be demonstration only or it may involve student interaction in selecting such variables as how many times the animation is shown and what the actual members of the data set are to be. Two experiments examined the question

of how interaction impacts on learning. This is an issue concerning control of and involvement in the learning process.

### 3.3.2 Dependent Variables

In general, the dependent variables in the experimental sequence were speed and accuracy on the post-test. In addition to these measures, student preference measures were taken to measure the attitudes of the subjects. Pretests of background knowledge were treated as covariates in the experiments.

## 3.4 XTango

The XTango animation package [59], was used to create the animations used in these studies and experiments. This software, developed by Dr. John Stasko, allows the creation of animations through XTango calls within a C program. It implements the path transition animation framework which he designed [58]. (XTANGO = X-window Transition-based ANimation GeneratiOn.) The animator can select the style of data representation. Basic images in the package are lines, rectangles, circles, ellipses, polylines, polygons, splines, bitmaps, text and compositions. The software also allows the use of numbers and text to accompany any data structures represented. XTango calls also allow highlighting and exchange of elements as well as the insertion of line style changes and color changes in the animations.

The XTango system can present a program in animated form in order to facilitate meaning and purpose. Animations are developed by first creating the graphical objects used to represent the data. These are created as images which may change in size, location, visibility, fill, or color as the animation progresses. Paths and transitions allow these changes to occur. An algorithm is mapped to the images and transitions by the use of an animation scene file.

In order to build an animation the animator should determine the algorithm operations to be demonstrated, add these operations to the source code file as XTango calls,

design the animation scenes to be used, and create the necessary linkages from operations to scenes.

XTango is available by anonymous ftp [55]. Some of the animations used in these experiments may be found in the subdirectory hf_studies.

## 3.5 Algorithms

Several algorithms were used in the experimental sequence. The first experiments were based on sorting algorithms. These algorithms were chosen because they are a basic of computer algorithm study. Many versions of the sorting algorithms exist and they present a varied range of difficulty. Information is often placed in alphabetical or numerical order to aid a reader in locating a target value. Because sorting has been widely studied by computer scientists, many techniques are available.

Later experiments were based on the Kruskal Minimum Spanning Tree algorithm. This algorithm was selected because graph algorithms form another important subclass of computer algorithms. In addition, this algorithm, unlike sorting which carries with it some intuitive knowledge gained from everyday experience, is generally unfamiliar to the participants. Thus, the experimental exposure provides the first contact with the concepts of the algorithm and the effects of the experiment can be more precisely judged. In addition, this algorithm manipulates a spatial two dimensional entity. Animations may have different effects on two dimensional objects than on the inherently one dimensional list of numbers. This algorithm is often presented in computer science classes which deal with algorithm analysis.

Kruskal's Minimum Spanning Tree Algorithm is applied to a weighted graph. In a weighted graph, weights are assigned to each edge to represent distances or costs. A minimum spanning tree of a weighted graph consists of

- a collection of edges,
- these edges connect all the vertices, and

- the combined weight of these edges is the minimum possible.

Kruskal's Algorithm for finding the minimum spanning tree of a graph proceeds in this manner:

1. First sort the edges by weight.
2. Select the shortest edge. Add it to the minimum spanning tree.
3. Select the next edge. Test to see if a cycle is formed. If no cycle is found, add the edge to the MST. Otherwise continue with the next edge.
4. Repeat step 3 until all nodes are connected by the minimum spanning tree.

## 3.6 Testing Software

In house testing software was developed for this experimental sequence. Using the X Window System and the Motif Widget Library, the software included an introductory screen giving instructions for use. Each successive screen presented one question and a set of possible answers. Answers were selected by use of a mouse. The mouse also controlled progress to the next question. This software allowed the subject to interact with the computer in entering answers to multiple choice or true/false format questions. The software recorded both answers and times for the post-test. This increased the efficiency of the analysis procedure and ensured no reaction of the experimenter to correct or incorrect responses.

A sample screen from the testing software appears in Figure 3.1.

Students were able to change answers while within the question, but were unable to return to a question once they had progressed beyond it. Each survey was designed to include questions which dealt with how to begin application of the algorithm, selecting the next step in the algorithm, concluding the algorithm, and key concepts of the algorithm.

In some of the experiments, a background process recorded the data sets subjects used. In addition, the process prompted the students to type in their reasons for selecting a particular data set for the animation. These answers were studied to try to understand how the animation aided the concept formation process.

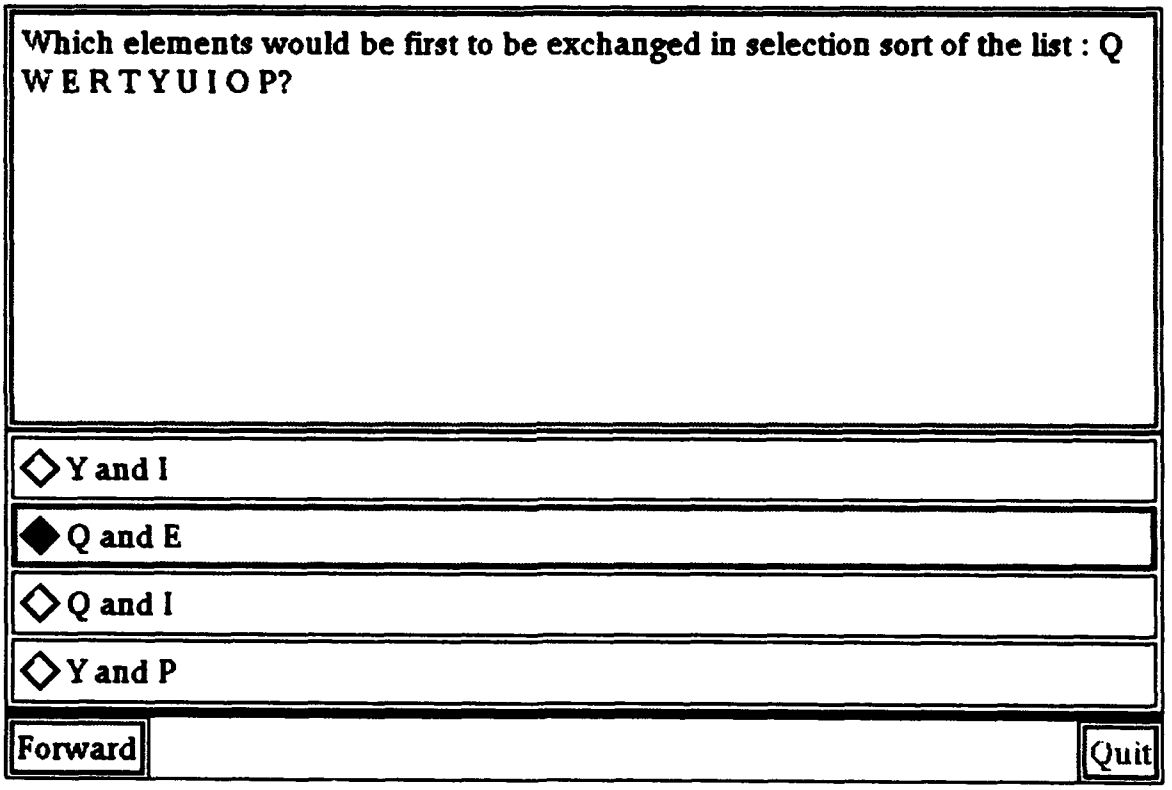



Figure 3.1: Testing Software Screen

## 3.7 Test Development

In order to be sure of the efficacy of a teaching intervention, we must find a means of measuring the learning and understanding which takes place. Anderson and Draper [2] argue that only the observations of real teachers in real classrooms have ecological validity. An alternative approach is to use pre and post-tests, along with formal experiments with control groups. Some evaluation measures are behavioral measures such as time to completion and error rate. Observation may be predetermined (such as time) or open-ended to allow for unexpected events.

Even when the method of measurement is ascertained, difficulties remain. The correct instrument must be selected. Selecting a question for a post-test is easy. However, designing a question to measure understanding is not so simple. One generally accepted method is to ask the learner to demonstrate measurable behavior designed to show achievement of a particular learning objective. Thus, behavioral objectives must be listed before the test is composed. Secondly, it is important that alternative presentations of the experimental material convey the same information. A third problem is analysis. Analysis must be planned to consider varied results such as performance and preference data as well as any covariates that can be determined.

These considerations point out the difficulty in asserting the value of the use of algorithm animation in the computer science classroom. Several measures must be gathered and more than one experiment must be run in order to overcome the obstacles and achieve convergent validity.

Several things were done to ensure that the tests would be appropriate for measuring understanding and application of the algorithm. Textbooks of the appropriate level were consulted for sample question format [53, 28]. Learning objectives were stated in behavioral form so that the questions would address key issues of the algorithm. A combination of format of question style was selected in order to provide a clear picture of learning. Multiple choice, true/false, short answer, and questions requiring the working out of the entire process of the algorithm were included. Questions were referred by college teachers of computer

algorithms courses.

Early experiments concentrated on a few on-line questions followed by opportunity for comment and a chance to work out one example of the algorithm. Later tests included a wider variety of test questions with more opportunity for short answer, working out of the entire algorithm, and free response answers. It was felt that this variety approach would allow a clearer picture of the knowledge level of the student. Some questions were designed to deal with procedural level topics while others were aimed at the operational level.

Pre-tests were used in several experiments. This was done in case the effect of prior knowledge would effect results. Pre-tests were based in the case of the sorting algorithm on basic concepts of following algorithmic processes. Pre-tests for the Kruskal Minumum Spanning Tree Algorithm were based on some basic graphic concepts and definitions [26].

# CHAPTER IV

# ESTABLISHING STUDENT PREFERENCES

## 4.1 Introduction

In our study of the use of animated algorithms, the first step was to establish student preferences and to determine whether these preferences match performance results. This chapter presents three related studies which examine the issue of student preference and data representation. Representations which are used are vertical bars, horizontal bars, and dots. The XTango algorithm animation package allows the selection of various representations of data which may be used to represent the same algorithmic procedure.

The elements on the screen represent data inputs. There are three representation of the data inputs, Vertical Bar, Dots, and Horizontal Bar. These versions are shown in Figure 4.1, Vertical Bar, Figure 4.2, Dots, and Figure 4.3, Horizontal Bar. For the Vertical Bar and Dot representations, the Y orientation displays the magnitude of the elements. For both bars, there is an extra cue for magnitude, the length of the bar. The Dot, however, only has one cue for magnitude, its height above the X-axis. The X dimension represents the current position in the array of numbers. For the Horizontal Bar representation, the X and Y indications are reversed. A change in array position is indicated by the movement of the element.

The first two studies sought to establish students preferences in terms of data representation and data labeling. The first study exposed students to three different data representations of the Quick Sort algorithm and gathered rankings of factors including clarity and best data representation. In the second study, twenty-one different views of Quick Sort were ranked by individuals. The third study was a formal experiment which investigated whether student preferences for a data set representation match performance. Data

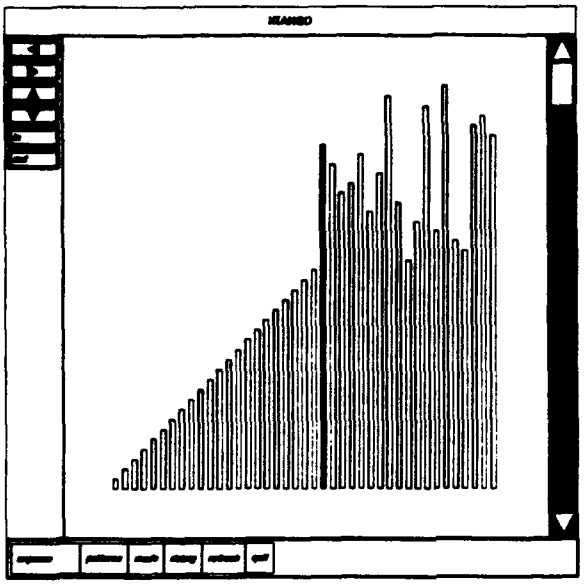


Figure 4.1: Quick Sort, Vertical Bar, Large Data Set

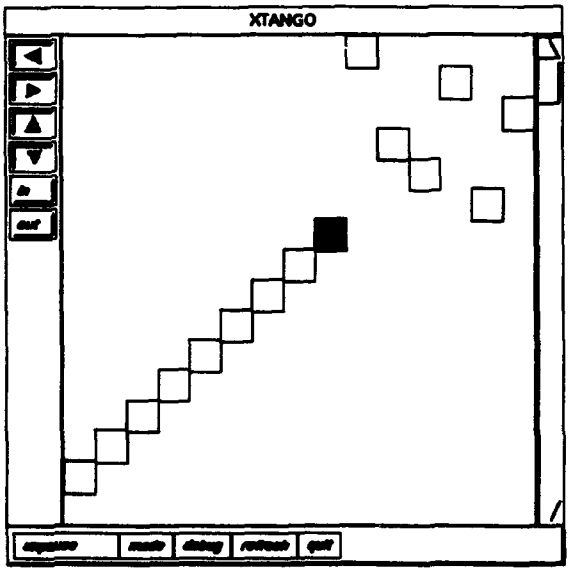



Figure 4.2: Quick Sort, Dot Representation

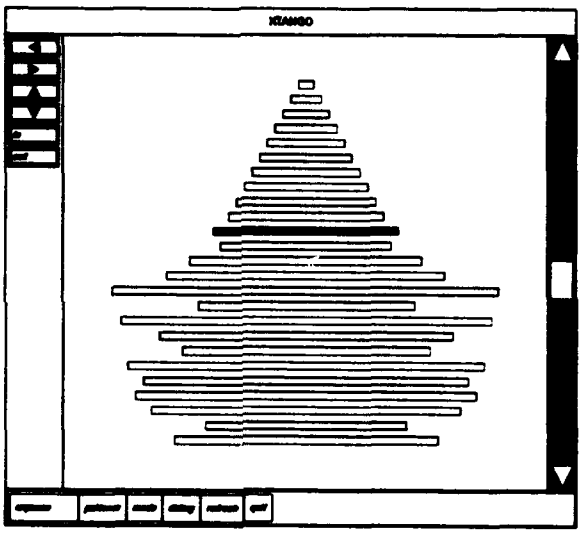


**Figure 4.3: Quick Sort, Horizontal Bar Representation**

set size was also examined in this experiment.

The task of research in the use of algorithm animations began with exploring student preferences. We also elicited students' insights on improving animations. Once these preferences were determined, we tested them to ascertain if they indeed resulted in more learning.

An informal study by Badre et. al [3] indicated indicated that students preferred the bar representations of data, while teachers tended to prefer the dot style of data representation. These teachers felt that the dot gave a better overall picture of the workings of the algorithm.

## 4.2 Exploratory Preference Study

### 4.2.1 Introduction

The focus of this study was to determine which of three representations was preferred by students in terms of speed, clarity, data representation, overall visual impact, best for new student, and other features. This was to be a comparative preference since each student would see all three versions.

### 4.2.2 Design

This was a 1 x 3 within subject design. The independent variable was data representation with three levels vertical, horizontal, and dot. Each group viewed all three representations but in a different order. Order was selected in a Latin Square method. All possible orders were not selected, but each representation appear first, last, and in the middle once. (VHD, DVH, HDV).

### 4.2.3 Subjects

Thirty-five students in a Computer Architecture class at the Georgia State University participated in the first study. These students were volunteers.

### 4.2.4 Materials

Three video tapes were prepared showing the Quick Sort algorithm in various representations. Each tape contained three representations. These tapes were recordings of a screen display of the XTango animations. Color and labels were not included in order to concentrate more fully upon the representation. The only value representation was height, width, or distance from the Y axis. Exchange of elements was visually indicated by the pair of elements traveling along curved paths from old to new location. The pivot element was represented as filled. When sorting was completed, the Vertical Bars were represented with the smallest in magnitude to the left of the screen, rising in height to the largest at the right of the screen. The Dot representation formed a slanting line from lower left to upper right of the screen. With the Horizontal Bars, the sorted array appeared as a pyramid, with the smallest element forming the top of the pyramid.

The Quick Sort algorithm was chosen because it was somewhat more difficult than some basic algorithms which might have been taught in high school. Students in the class had been taught sorting methods in a previous Data Structures course.

### 4.2.5 Procedure

The class met in classrooms where equipment permitted the videotape to be shown on a large screen. Each student was randomly assigned to one of three groups, Vertical-Horizontal-Dot, Dot-Vertical-Horizontal, and Horizontal-Dot-Vertical. Each group viewed three representations of the data shown on a prepared videotape.

Students had been taught the Quick Sort algorithm in a previous class, and were given a handout (see Appendix C) to review the process before viewing the animations. They were told that the animation would be of the Quick Sort process, but were given no information as to how the data would be represented or how the modeling would be carried out. As each animation finished, students were asked to rate that version of the animated algorithm for speed and clarity. Subjects in the study were asked to respond to the questions shown in Appendix B. Some of the questions were the following:

Table 4.1: Exploratory Study

| | Preferred Version | Aided Understanding | New Student |
|---|---|---|---|
| Dots | 1 | 1 | 2 |
| Horizontal | 16 | 16 | 15 |
| Vertical | 18 | 18 | 19 |

- What did you think about the speed of the animation?
- How well did you understand what each bar or dot represented?

After all three versions of the animation were shown, students were asked to compare the three representations. Factors included were personal preference, best version to use for teaching a new student, which most aided their understanding, and best representation of the data. They were also asked to compare the versions for clarity and speed.

### 4.2.6 Analysis

Both vertical and horizontal bars were much preferred over the dot representation ($p < 0.05$). Results indicated that there was no clear preference between the vertical and horizontal bar versions of the animation. These preferences appear in Table 4.1. The vertical representation received the highest number of best votes in all categories. However, this difference was not statistically significant above the horizontal bars. Analysis of the preference data indicated no effect of order.

Given a ranking scale of 1 to 5, students were asked to rate the speed of the three representations. For example, in speed 5 stood for Too Fast, 4 represented A Little Fast, 3 was Just Right, 2 was A Little Slow, and 1 was Much Too Slow. With regard to speed, dots averaged 3.8, Horizontal averaged 3.2; and Vertical averaged 2.8 . This suggests that the dot representation seemed fast, the horizontal a little too fast, and the vertical a small fraction too slow. This may have affected the preference scores for the dot representation, since it was viewed as too fast.

The Quick Sort algorithm was presented three ways: dots, vertical bars, and horizontal bars. For each of the criteria below, subjects were asked to select the best and worst representation. Results of the questions appear in Table 4.2.

| | Best | Worst |
|---|---|---|
| Speed | _____ | _____ |
| Color | _____ | _____ |
| Clarity | _____ | _____ |
| Overall Visual Impact | _____ | _____ |
| Data Representation | _____ | _____ |

Not all sums are equal to the number of subjects. Some students answered a question with two choices; others omitted some answers.

$\chi^2$ tests (See Table 4.3 for a summary) indicate that vertical and horizontal are preferred to dots in all categories. Vertical representation is preferred for clarity and receives the highest number of best votes for speed, color, overall visual, and data representation. This seems to support our intuition that the vertical is the favored choice. This may be due to added familiarity with this representation since bar graphs are frequently used to present numeric data. This version is closely followed by the horizontal representation. Dots received the majority of worst characteristic selections.

Students preferred the two representations in which the orthogonal factors, i.e. size of an element and location in an array, were coded in distinctly different fashions. Comments made indicated that the dot representation was initially confusing as to what was indicated by the X and Y dimensions of the individual dots. This seems to be because the possibility existed that the X dimension could be the value. No visual evidence was included to clarify this questions. In fact, some students expressed verbal puzzlement over the second dimension. Both the vertical and the horizontal bar representations, however, were readily understood by the subjects. A few subjects made remarks as they watched such as "The height is the value," thus confirming that the coding was apparent.

Student response to the dots was generally negative, although many researchers believe that the dots are the more effective method for presentation of this type of sorting algorithm. Since the visual coding was not explained to the students, they were unable to

Table 4.2: Best and Worst Characterizations

| | | Best | Worst |
|---|---|---|---|
| Speed | | | |
| | Dots | 4 | 13 |
| | Horizontal Bars | 11 | 1 |
| | Vertical Bars | 16 | 5 |
| Color | | | |
| | Dots | 4 | 11 |
| | Horizontal Bars | 11 | 3 |
| | Vertical Bars | 16 | 2 |
| Clarity | | | |
| | Dots | 4 | 16 |
| | Horizontal Bars | 8 | 2 |
| | Vertical Bars | 14 | 3 |
| Overall Visual | | | |
| | Dots | 2 | 27 |
| | Horizontal Bars | 15 | 0 |
| | Vertical Bars | 17 | 3 |
| Data Representation | | | |
| | Dots | 2 | 27 |
| | Horizontal Bars | 14 | 1 |
| | Vertical Bars | 18 | 3 |
| Total Votes | | | |
| | Dots | 16 | 94 |
| | Horizontal Bars | 59 | 7 |
| | Vertical Bars | 91 | 16 |

Table 4.3: Best Ratings Comparisons

| Difference of Vertical and Horizontal Bars from Dots | | | |
|---|---|---|---|
| Characteristic | $\chi^2$ | d.f. | p |
| Preferred Version | 17.3 | 2 | .05 |
| Aided Understanding | 17.3 | 2 | .05 |
| Best for New Student | 16.18 | 2 | .05 |
| Speed | 7.06 | 2 | .05 |
| Color | 7.06 | 2 | .05 |
| Clarity | 5.80 | 2 | .06 |
| Overall Visual | 13.19 | 2 | .05 |
| Data Representation | 13.77 | 2 | .05 |

easily map the representation to their conceptual idea of the algorithm. This result indicates that with the dot representation some verbal or written description of the meaning of the representation is needed to allow a fuller appreciation of what is being shown. Students were also asked to comment on the use of animations to teach algorithms. Several comments made by students indicated that:

- Students would like animations used to teach algorithms.
- Students believe that animations can be helpful in the learning process.
- Several students indicated a desire to have labels or text added to the animation.

These comments were used in designing the experiment presented later in this chapter.

## 4.3 Ranking Study

The previous study indicated some trends concerning student preferences. The next step was to follow up on the student suggestions for adding labels. In this study, labels were used to indicate the value of data elements. These were numeric values which appeared at the top, bottom, left, right, or in the middle. Students were asked to rank twenty-one different representations of the data according to individual preference.

### 4.3.1 Design

Order of the materials was randomized to avoid order effects. Each subject was to rank the views according to individual preference.

### 4.3.2 Subjects

Subjects were 26 students in a Computer Architecture Class at the Georgia State University. They had taken part in the study described previously. This study occurred approximately two weeks after the first study.

### 4.3.3 Materials

Twenty-one different representations of a sorting algorithm were prepared, as summarized in Table 4.3.3.

Features of interest were:

- Filled (solid black) or empty (black outline)
- Label or no label
- Location of label

Nine of these were vertical bars, nine were horizontal bars, and four were dots. Of the vertical bars, three were solid, and each had one of the following labeling conditions: no label, label top outside and label bottom outside. The six empty vertical bars featured one of the following labeling conditions: no label, top inside, top outside, bottom inside, bottom outside, and center. Labeling conditions for the horizontal bars are similar, except that top becomes left and bottom becomes right. There were also two solid and two empty dots, either labeled in the center or containing no label. Dots were included to confirm previous results which placed them last in students preference.

### 4.3.4 Procedure

The twenty-one views were printed and duplicated. The students were asked to shuffle the views in the order of their preference.

### 4.3.5 Analysis

Results of this study which appear in Table 4.5. showed that:

- Students ranked vertical bars higher than horizontal.
- Among vertical bars, solid bars were ranked higher.
- The highest ranked horizontal bar is empty with a center label.

Table 4.4: Twenty-one Views

| | Filled | Empty |
|---|---|---|
| Vertical | Label Top Outside | Label Top Outside |
| | | Label Top Inside |
| | Label Bottom Outside | Label Bottom Outside |
| | | Label Bottom Inside |
| | No Label | No Label |
| | | Label Center |
| Horizontal | Label Right Outside | Label Right Outside |
| | | Label Right Inside |
| | Label Left Outside | Label Left Outside |
| | | Label Left Inside |
| | No Label | No Label |
| | | Label Center |
| Dots | | Label Inside |
| | No Label | No Label |

Table 4.5: Preferences for Views: Views included received at least one vote for number one and occurred in the top ten more than ten times.

| Object Description | Fill | Label Placement | Number One Frequency | Top Ten Frequency |
|---|---|---|---|---|
| Vertical Bars | Filled | Top Outside | 5 | 19 |
| Vertical Bars | Filled | Bottom Outside | 4 | 19 |
| Horizontal Bars | Empty | Center | 3 | 16 |
| Vertical Bars | Filled | None | 2 | 17 |
| Vertical Bars | Empty | None | 2 | 17 |
| Vertical Bars | Empty | Bottom Inside | 2 | 15 |
| Horizontal Bars | Filled | Left Outside | 2 | 14 |
| Horizontal Bars | Filled | None | 2 | 14 |
| Horizontal Bars | Filled | Right Outside | 2 | 13 |

- Although the highest ranked horizontal bar is empty, all other horizontal bars appearing in the top ranked group were solid.
- For solid vertical bars, favored label placement is top outside or bottom outside.
- For empty vertical bars, no label or bottom inside label were ranked highest.
- For solid horizontal bars, label placement of left or right outside or no label were ranked approximately at the same level.

These results appear in Table 4.5. This led to the hypothesis that in sorting algorithms students would prefer solid vertical bars with labels at either top or bottom of the bar. Horizontal bars would be the next preference, with the dot representation being the least desired.

Several findings resulted:

1. Most commonly chosen as first preference is the solid vertical bar representation with labels at the top of each bar. Second ranked was the solid vertical bar representation with labels at the bottom.

Table 4.6: Summary of Preference Results

| Factor | Number One Frequency | Top Ten Frequency |
|---|---|---|
| Vertical Bars | 16 | 153 |
| Horizontal Bars | 9 | 101 |
| Dots | 1 | 6 |
| Solid | 17 | 98 |
| Empty | 9 | 162 |
| Labels | 20 | 200 |
| No Labels | 6 | 60 |

2. In all, solid vertical bars were ranked first 11 times, solid horizontal bars 6 times, and empty vertical bars 5 times, and empty horizontal bars 3 times.

3. Most often ranked in the top 10 was black vertical bar, labels at the bottom and solid vertical bar, top labels.

Table 4.6 shows the number one frequencies and the top ten frequencies for types of representation, solid and empty, and label or no label. We observe that labels are strongly preferred to no labels, while vertical bars are chosen many more times than horizontal bars or the dot representation. Again, the dot is the least favored.

Further analysis ranked each view by assigning a point value to the ranking and summing the rankings. For example, a first choice received 20 points, a second choice 19 points, and so on to last choice with 0 points. The top five views were as follows:

1. Vertical, empty, label top outside, sum of rankings: 438

2. Vertical, filled, label top outside, sum of rankings: 368

3. Vertical, filled, label bottom outside, sum of rankings : 365

4. Vertical, empty, label top inside, sum of rankings : 330

5. Vertical, empty, label bottom inside, sum of rankings : 322

Analysis of variance with factors data representation and fill style indicated a significant difference with respect to representation ($F = 47.50$. d.f. 20, $p < 0.01$). Pairwise t-tests showed that vertical bars were preferred to horizontal bars ($p < 0.05$) and to dots ($p < 0.01$), while horizontal bars were also preferred to dots ($p < 0.05$). All the top six were vertical bars with labels, indicating student preference for these parameters.

### 4.3.6 Discussion

Overall, results indicate that students prefer vertical bars with numeric labels. These two studies show clear results in the case of student preferences. Students tend to prefer data representation for sorting algorithms which:

- Are vertical bars
- Show labeled values

In addition to this preference, students request the use of text or vocal explanations to accompany the animation. This is the one suggestion which occurs most frequently. Exploration of this factor occurs in Chapter 7. Other suggestions concerning the use of color to serve as indicators of various steps are also discussed in that chapter. Once these preferences were established, the next step was to investigate whether these preferences were intuitively correct in determining the best format for data representation. The next experiment compared three data representations to determine which was best for use with this algorithm and to also determine whether student preferences matched or contradicted student performance.

## 4.4 Do Preferences Match Performance?

The previous studies examined student preference. However, the representation preferred may not be the representation which is most helpful in learning the algorithm. The preference studies indicated a desire by the students for vertical bars with labels. This was examined in two experiments. The first experiment, which follows, dealt with the issue of

data representation. Data labels were not included in this experiment in order to avoid the confounding of the effects of labeling and representation. The second experiment explores the labeling issue is explored in Chapter 5.

If preferences for representation type indeed match performance, students would be expected to show highest scores on the vertical bar representation, closely followed by the horizontal bar representation. Dots would be expected to place solidly last. Performance was measured by scores and time data on a post-test given immediately following exposure to the animation. In order to take a closer look at the effects of representation on performance, a two-factor experiment was conducted at the Georgia Institute of Technology. The first factor was the representation of the data. Three representations were included: vertical bars, horizontal bars, and dots. Another issue of concern was optimal data set size for the animations. Conflicting hypotheses were that large data set size:

1. Presented a clearer and more detailed overall picture.

2. Was too distracting and confusing.

This experiment provided more information as to whether representation and data set size did, indeed impact student understanding of the algorithm.

### 4.4.1 Design

This was a 3 x 3 between-subject design shown in Table 4.7. The first factor was data representation style: vertical bar, horizontal bar, and dot. The second factor was data set size, with small being characterized as less than 10 data elements, moderate characterized as 10 to 40 data elements, and large characterized as more than 40 data elements. Dependent variables analyzed were completion time and accuracy on the post-test. Although the large size data set represented here was not as large as a 100 or 200 data set, this size was selected as large enough to give an overall picture of the algorithm while not overcrowding the screen. It was felt that too high a screen density would interfere with comprehension as well as preclude the addition of legible numeric labels in later experiments. It was expected that vertical bar format would lead to faster times and greater accuracy on the post-test.

The dot representation was expected to lead to the lowest scores. Medium data set size was predicted to be the optimal choice as measured by accuracy and time on the post-test.

### 4.4.2 Subjects

Subjects were thirty-six students in CS2201, a Data Structures and Algorithms class at the Georgia Institute of Technology. Students were volunteers who received class credit for participation in the experiment. These students were chosen because the algorithm to be used was scheduled to be presented during the class. The experiment was conducted during the week before the algorithm was taught.

### 4.4.3 Materials

Three versions of the Quick Sort Algorithm were prepared. These animations represented each of the three types previously described: vertical bar, horizontal bar, and dots.

Data sets were prepared for the small, medium, and large conditions containing 9, 25, and 41 elements respectively. Larger data sets were not used because of factors of screen density and the difficulty of inserting legible numeric data labels.

A text description of the algorithm to be used, Quick Sort, was prepared. This algorithmic description was based on textbooks at the level of the class. Two instructors who commonly taught this algorithm examined it for clarity.

The post-test questions were based on questions in text books which covered the quick sort algorithm. An on-line survey form was created which allowed these questions to be presented to each subject on the computer screen. Questions were presented in multiple choice or true/false format. Responses and times were recorded by the computer in log files for later analysis.

### 4.4.4 Procedure

Each student completed a consent form and a pretest. Students were randomly assigned to one of nine groups based on data set size (small, medium, large) and representation (vertical bars, horizontal bars, and dots). They then were asked to read a handout on the algorithm.

Table 4.7: Design of Experiment

| | REPRESENTATION TYPE | | |
|---|---|---|---|
| SIZE | Vertical Bars | Horizontal Bars | Dots |
| Small | | | |
| Medium | | | |
| Large | | | |

This handout was based on text materials at a level intended for the course in which they were enrolled. They were told that they would view an animated version of the algorithm. Students then viewed the animation. The representation was not explained to them. They were allowed to view it until they felt ready for the post-test. Students then completed a computerized post-test as well as working out an example of the algorithm on paper.

### 4.4.5 Analysis

ANOVAs for time and accuracy did not reveal a significant difference between groups according to either data set size or representation. Cell means for accuracy are found in Table 4.8; while cell means for time appear in Table 4.10. See Tables 4.9 and 4.11 for ANOVA results.

Comments made by students indicated a desire to see animated algorithms added to the teaching of algorithms. Several students inquired whether they could access other algorithms as they were presented in class.

Suggestions for improvements to the animations included labels and some indication of the current location of the pivots, a crucial feature of the Quick Sort algorithm.

### 4.4.6 Discussion

No significant difference in accuracy was found due to representation or data set size. The percentage correct of the seven questions in the post-test was 64% for the Vertical Bars, 68% for the Horizontal Bars, and 74% for the Dots.

Table 4.8: Accuracy Results: Number Correct of Seven

| CELL MEANS | | | | |
|---|---|---|---|---|
| SIZE | Vertical Bars | Horizontal Bars | Dots | Marginal Means |
| Small | 4.50 | 5.25 | 4.75 | 4.83 |
| Medium | 4.50 | 4.50 | 6.00 | 5.00 |
| Large | 4.50 | 4.25 | 4.75 | 4.50 |
| Marginal Means | 4.50 | 4.67 | 5.17 | |

Table 4.9: Accuracy Results

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 821.77778 | 1 | 821.77778 | 672.36 | 0.0000 |
| REP | 2.88889 | 2 | 1.44444 | 1.18 | 0.3221 |
| SIZE | 1.55556 | 2 | 0.77778 | 0.64 | 0.5370 |
| RS | 4.77778 | 4 | 1.19444 | 0.98 | 0.4363 |

Table 4.10: Cell Means for Time

| CELL MEANS | | | | |
|---|---|---|---|---|
| SIZE | Vertical Bars | Horizontal Bars | Dots | Marginal Means |
| Small | 150.3 | 143.5 | 194.0 | 162.6 |
| Medium | 160.3 | 137.8 | 116.0 | 138.0 |
| Large | 134.5 | 169.3 | 122.3 | 142.0 |
| Marginal Means | 148.3 | 150.2 | 144.08 | |

Table 4.11: ANOVA of Time Results

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 755632.01190 | 1 | 755632.01190 | 467.46 | 0.0000 |
| REP | 226.72126 | 2 | 113.36063 | 0.07 | 0.9324 |
| SIZE | 4134.69828 | 2 | 2067.34914 | 1.28 | 0.2953 |
| RS | 13762.34409 | 4 | 3440.58602 | 2.13 | 0.1058 |
| ERROR | 42027.66667 | 26 | 1616.44872 | | |

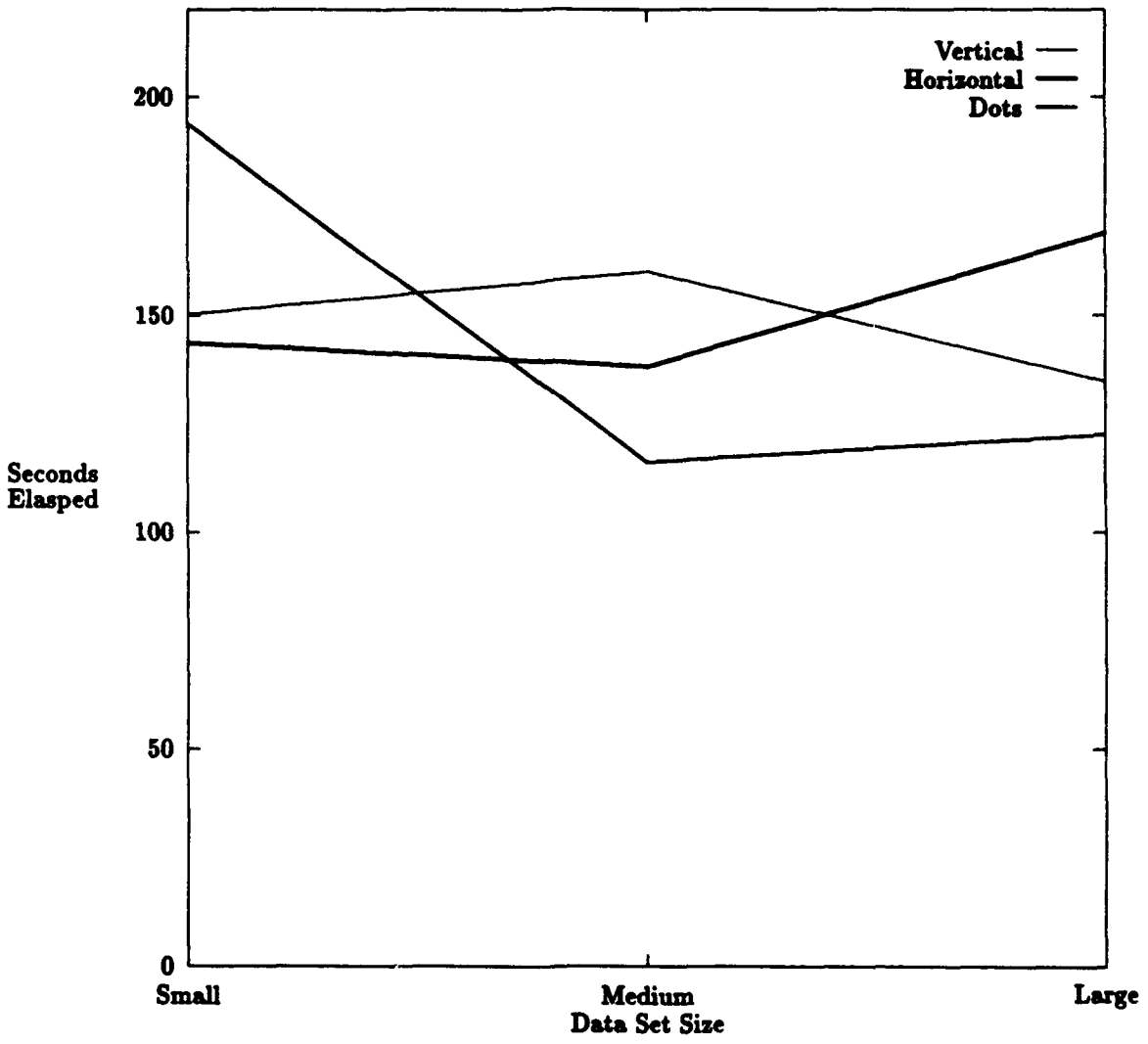



Figure 4.4: Results of On-line Post-Test

An interaction between size and representation was found in the data analysis. This indicated that horizontal bars in the large data set size were worse than dots in the medium data set size. However, closer examination of the data revealed an outlier. This outlier was an unusually large time value for the post-test. It was due to one subject who seemed reluctant to commit to any response. When the outlier was removed, the interaction no longer appeared. The time results are graphed in Figure 4.4.

Although there were no main effects for either representation or data set size in this experiment, the results of the experiment proved valuable. It drew attention away from representation of data elements to other aspects of the animation. Students' numerous suggestions derived from this experiment helped chart the direction of future studies by providing many useful ideas for the course of future experimentation. Students made suggestions about improving the animation. The following represent students' suggestions, with the first six being those most often given:

1. Use labels on the data elements (suggested several times). These labels would be numbers which would indicate the relative value of the data elements when the data elements were numbers. Such labels could appear above, below, or superimposed on the data elements. In other algorithms such as graph based algorithms, these labels might represent weights on edges or the order in which nodes were visited. In any case the label must be relative to the algorithm.

2. Emphasize the pivot by making it a different color or label it. The pivot is a key element of the Quick Sort Algorithm. It is used in every comparison and divides the elements of less value than the pivot from the elements of greater value. The pivot changes at each recursive call of the algorithm.

3. Include some indication of the current data range. Possibilities include separation by lines, different colors, and arrows marking the borders.

4. In Quick Sort, show some indicator of the two pointers which are used to determine items to be swapped. Include moving pointers or arrows for the up and down indi-

cators. Alternately, color the two bars where up and down are currently a different color.

5. Add a textual commentary of the process of the algorithm. This textual commentary would parallel the visual progress of the animation.

6. Show lines of code or pseudo code corresponding to animation. Several students wished to see Pascal, C, or pseudo code corresponding to each portion of the animation as the animation was displayed.

7. Use greater than/less than signs to show progress of pointers. The greater than signs could indicate the pointer which searches in the Quick Sort Algorithm for the items greater than the pivot.

8. Indicate comparison with colors instead of flashing the bars being compared. Some students found the flashing of the bars to be distracting.

9. Give the user step-by-step control. Students wished to be able to pause the animation at crucial points and examine the representation more closely.

10. Provide an explanation of the representation. Students wished to have explained what the data elements represented, for example in the dot representation the X coordinate represented position in the array while the Y coordinate represented value of the data element.

11. Add an auditory commentary. This commentary would provide additional information about the progress of the animation.

Suggestions 1 through 4, 7 and 8 dealt with visual representation of graphical elements. Student suggestions indicated that the key variables of the algorithm, i.e. the pivot point and upper and lower bounds, needed to be emphasized graphically. Note that some of these variables were not coded visually in the experiment (such as the upper and lower bounds of the current data set). Furthermore the graphical encoding techniques suggested

Table 4.12: Student Preferences

| Preference Results | |
|---|---|
| Representation | Vertical Bars |
| Labels | Numeric labels |
| Color | As indicators |

fell into two categories: modifying the object already on the screen as in colors or labels and adding new elements such as arrows or greater than/less than signs. The suggestion of new elements indicates that subjects were able to learn enough about the algorithm to identify critical elements.

Suggestions 5, 6, 10 and 11 concerned adding textual or verbal description to the animation. This indicates that subjects would like further explanation of the process they are viewing and that they did not already feel that they were overloaded with information. The request for code probably reflected the programming nature of the class from which the students were drawn.

The last category of suggestion was step-by-step control by the user. This seems to indicate that students perceived control as an important facet of learning. Some information was lost because it was removed before the subject had fully processed it.

From these experiments we learn much about student preferences. As illustrated in Table 4.12, students prefer bar representations of the data, labels are desired, and that optimal data set size interacts with representation. Thus, this experiment was the source of several guides to continuing the course of investigation.

These studies of students preferences established some clear choices in the question of data representation. Since no advantage was demonstrated for any representation in terms of accuracy, student preferences can, indeed, help to serve as guides in choices for developing the animations.

The next area of exploration was design of the animations themselves. Design issues

which we explored include the use of data labels, interactive or passive data sets, and the use of color cues and text commentary to mark important parts of the algorithmic process. Chapter 5 explores the use of data labels in these animations while Chapter 6 contrasts active and passive design features. Labeling of important features of the algorithmic process is explored in Chapter 7.

# CHAPTER V

# LABELING DATA ELEMENTS

## 5.1 Introduction

The next step in the research was to explore design issues connected with the building of the animations and the nature of the tasks these animations may be best applied to helping. One such design issue was the question of the labeling of data elements with numeric values.

In this chapter we examine the effect of labeling data elements with numeric values. Work by Glucksberg [25] on Duncker's candle problem suggests that labels may play an important part in problem solving. Preference studies show that the labeled versions are preferred to unlabeled versions at a ratio greater than three to one. During the previous experiment on data representation and data set size, students were asked to suggest ways to improve the animation. They were told that their suggestions would be used to provide a basis for later versions of the animations and asked to base their suggestions on improvements which would increase understanding of the algorithm. Students most often suggested that labels would improve the animation. This factor warranted a formal investigation.

Those suggestions made most frequently were to label the data, emphasize key elements by color changes, include some indication of the subset of the data structure currently under consideration, and the addition of text in the form of explanation, pseudo-code, or actual code.

This experiment examined the effect of labeled data versus non-labeled data versions of Selection Sort and Quick Sort XTango animations on accuracy and time to complete the post-test. This served to investigate whether student preference was a reliable guide to animation design or whether preference is not an accurate indicator of the most effective presentation. The experiment compared groups using labeled and non-labeled data sets.

The Labeled groups saw the algorithm with each element labeled by value. The Non-labeled group judged the value of the data elements in the vertical bar representation by the height of the bars, while in the dot version, value was represented by the Y-coordinate of the dot. Two data representations were used. They were vertical bar (preferred by students) and dots.

The second part of this experiment focused on the task issue. Two algorithms of different complexity were used in the experiment as a second factor in order to determine if animations were more effective in teaching simpler or more complex algorithms. Quick Sort is $O(n \log n)$ while Selection Sort is $O(n^2)$. Quick Sort is generally perceived as being more difficult to learn than Selection Sort, but more efficient in operation. One question of concern was whether there would be different effects for different tasks and whether order of presentation would affect results.

The experiment was conducted at the Georgia Institute of Technology using the XTango algorithm animation package. Results indicated that labeling the data did not improve accuracy on the post-test.

## 5.2 Experiment: Do Labels Help?

How much influence does the presence or absence of numeric labels have in the effectiveness of animated algorithms in teaching computer algorithms?

**Hypothesis**: Numeric labels on data sets in animated algorithms aid in understanding and application of the algorithm.

It was hypothesized that those students who viewed the labeled versions of the algorithm animation would achieve a higher degree of understanding of the algorithm as measured by higher scores on the post-test and quicker times to complete the post-test. Since the labels provide additional information, students should be better able to understand the steps of the algorithm as represented by the animation.

How much influence does the representation first seen (vertical bars or dots) have on the effectiveness of animated algorithms in teaching computer algorithms?

Table 5.1: Design of Data Element Labeling Experiment

| LABELING ISSUES | | | |
|---|---|---|---|
| Seen First | | Labeled | Unlabeled |
| Quick Sort | Selection Sort Results | | |
| | Quick Sort Results | | |
| Selection Sort | Selection Sort Results | | |
| | Quick Sort Results | | |

**Hypothesis**: The animation first seen will affect results on the post-test.

It was hypothesized that students who viewed the vertical bar animation first would achieve a higher degree of understanding of the algorithms since these students will have a more familiar first introduction to learning sorting algorithms with the animation system.

## 5.3 Design

This was a 2 x 2 (within 2) design, as illustrated in Table 5.1. One variable was Label or No-label condition; the second variable was the algorithm first seen; the within subject variables were the algorithm and the representation, since each participant saw both algorithms and both vertical bars and dots.

## 5.4 Subjects

Subjects were forty students at the Georgia Institute of Technology who were taking CS1410, the first programming course for Computer Science majors at the Institute. This course is also required for most Engineering majors at Georgia Tech. These students were volunteers who received class credit for their participation.

## 5.5 Materials

Animated versions of the two sorting algorithms were prepared. Numeric labels were used to represent the value of the data elements. In one pair of animations, vertical bars were

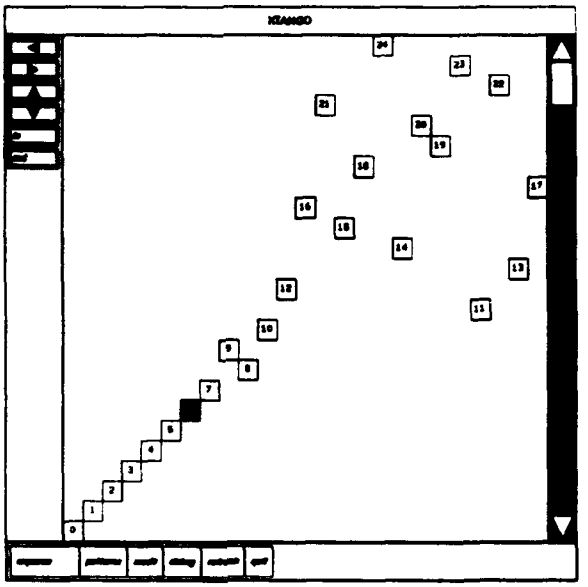


Figure 5.1: Quick-Sort, Dots, Labels

used for the sort algorithms. For the second pair of animations, the dot representation was used. Labels representing the value of the data element appeared within the bar or dot. In the Selection Sort, data was represented as empty bars when unsorted, and became solid bars once they were in position. In the Quick Sort, the pivot element was represented as solid, while other elements were represented as empty. Comparison was represented by blinking of the compared pair of data elements. Examples of the four conditions appear in Figures 5.1, 5.2, 5.3, and 5.4.

Students in various groups saw either labeled or non-labeled versions of both animations. One algorithm was represented by the dot format and the other by vertical bars. Students were randomly assigned to conditions. Order and representation of the algorithms presented was alternated randomly to avoid order effects.

An on-line post-test consisting of ten multiple choice or true/false questions for each algorithm was prepared. See Appendix D. The on-line nature of the post-test allowed accuracy and time to be automatically recorded for each subject. The Paper Fold Test and a Vocabulary Test were given so that verbal and spatial ability could be analyzed as covariates.

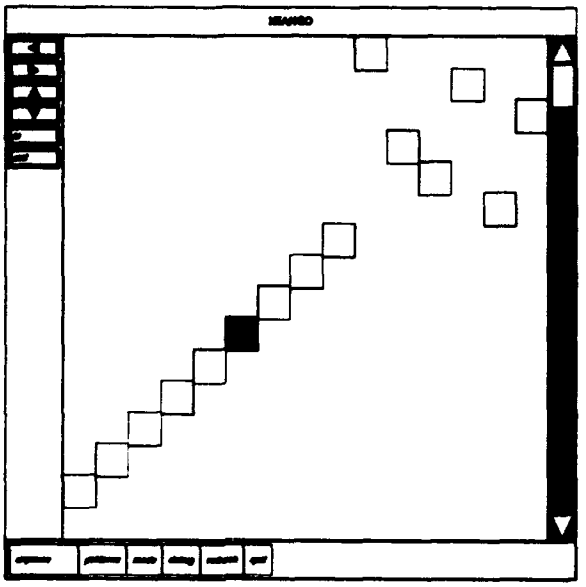


Figure 5.2: Quick-Sort, Dots, No-Labels

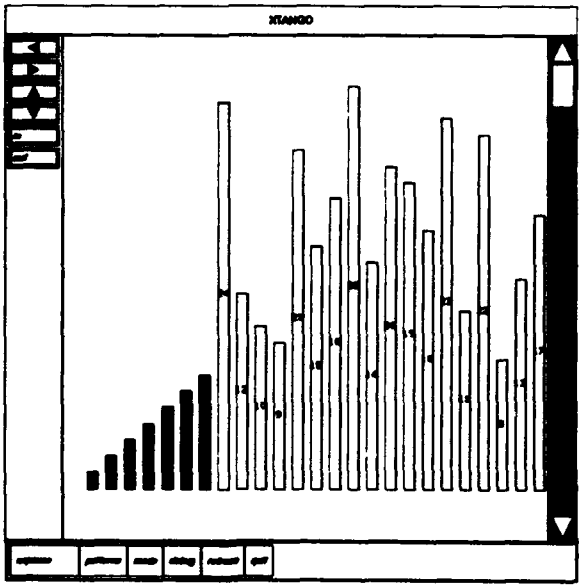


Figure 5.3: Selection Sort, Vertical Bars, Labels

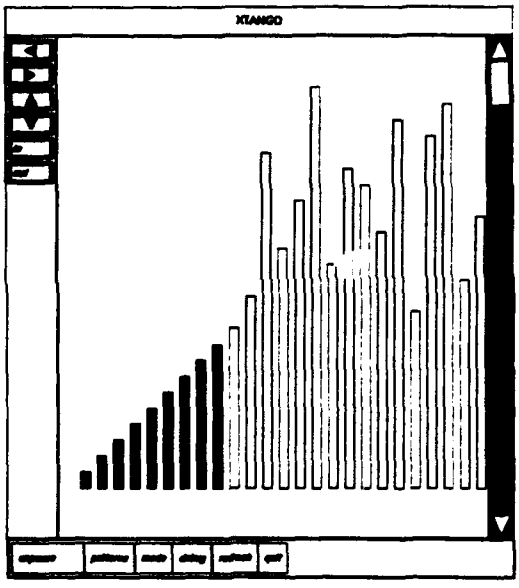


Figure 5.4: Selection Sort, Vertical Bars, No Labels

## 5.6 Procedure

Each student completed a consent form and a pretest of computer knowledge and exposure. The Paper Fold Test and a Vocabulary Test were given. The first algorithm was presented briefly by a printed description at the beginning of a laboratory session. The Quick Sort and Selection Sort descriptions appears in Appendix F. The XTango animation followed. A post-test of understanding and application of the algorithm immediately followed the demonstration. The procedure was repeated with the second algorithm. Each student saw both the Quick Sort and the Selection Sort algorithms. One sort algorithm used the vertical bar representation while the other used the dot representation. Each subject, therefore, saw the algorithms and representations in a balanced fashion.

Students were asked to give their preference as to which representation they preferred.

## 5.7 Analysis

Dependent variables were accuracy and time on the post-test. This test was designed to measure behavioral objectives based on concepts and applications of the sorting algorithms. Covariates of spatial and verbal ability (as measured by the Paper Folding Test and a Vocabulary Test) were also analyzed. These covariates were not found to be significant and were dropped from further analysis.

Table 5.2: Time Results In Seconds

| Time Results | | | | |
|---|---|---|---|---|
| Quick Sort | | | | |
| | | Quick First | Selection First | Marginal |
| | Labeled | 516.0 | 596.0 | 556.0 |
| | Unlabeled | 412.2 | 482.2 | 447.2 |
| Selection Sort | | | | |
| | Labeled | 302.4 | 416.3 | 359.5 |
| | Unlabeled | 335.5 | 393.9 | 364.7 |

Table 5.3: ANOVA, Time, Quick Sort Post-test

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| MEAN | 64102.40000 | 1 | 10064102.40000 | 261.32 | 0.0000 |
| LABEL | 18374.40000 | 1 | 118374.40000 | 3.07 | 0.0881 |
| FIRST ALG. | 56250.00000 | 1 | 56250.00000 | 1.46 | 0.2347 |
| LA | 250.00000 | 1 | 250.00000 | 0.01 | 0.9362 |
| ERROR | 1386429.20000 | 36 | 38511.92222 | | |

Analysis of variance was conducted. This method of analysis was selected in order to evaluate the effect of each factor and any possible interactions between factors. Factors used in the first analysis were Label or No/Label condition and algorithm first seen. Analysis was also conducted of using the factor of representation first seen. No significant difference was observed for the representation, so these groups were collapsed for future analysis.

Cell means for time on both algorithms are shown in Table 5.2. Analysis based on elapsed time for the post-test disclosed that time on the Post-test was marginally significant for the labeled condition with the Quick Sort ($F=3.07$, d.f.=1,36, $p < 0.09$). Those who viewed the labeled representation took slightly longer to complete the post-test. These ANOVA results appear in Table 5.3. No difference was observed as a result of labeling or of algorithm first seen for time for the Selection Sort. These ANOVA results appear in Table 5.4. However, an effect of algorithm first seen was observed for the Selection Sort algorithm. Further study of these results disclosed the fact that those who saw the Selection Sort first took longer on the Selection Sort Post-test.

Scores were not significantly different in either accuracy or time for the group which viewed the vertical bars first as compared to the group which viewed the dot representation first.

No effect of labeling or algorithm first seen was found for the Quick Sort or for the

Table 5.4: ANOVA Time, Selection Sort Post-test

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| MEAN | 5242484.02500 | 1 | 5242484.02500 | 243.88 | 0.0000 |
| LAB | 286.22500 | 1 | 286.22500 | 0.01 | 0.9088 |
| FIRST ALG | 74218.22500 | 1 | 74218.22500 | 3.45 | 0.0713 |
| LA | 7700.62500 | 1 | 7700.62500 | 0.36 | 0.5532 |
| ERROR | 773851.90000 | 36 | 21495.88611 | | |

Table 5.5: Accuracy Results: Number Correct Out Of Ten Questions

| Accuracy Results | | | | |
|---|---|---|---|---|
| Quick Sort | | | | |
| | | Quick First | Selection First | Marginal |
| | Labeled | 5.5 | 5.7 | 5.60 |
| | Unlabeled | 5.7 | 5.3 | 5.50 |
| Selection Sort | | | | |
| | Labeled | 7.5 | 7.1 | 7.30 |
| | Unlabeled | 7.2 | 7.7 | 7.45 |

Table 5.6: Accuracy on Quick Sort Post-test

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| MEAN | 1232.10000 | 1 | 1232.10000 | 548.96 | 0.0000 |
| LABEL | 0.10000 | 1 | 0.10000 | 0.04 | 0.8340 |
| FIRST ALG | 0.10000 | 1 | 0.10000 | 0.04 | 0.8340 |
| LA | 0.90000 | 1 | 0.90000 | 0.40 | 0.5306 |
| ERROR | 80.80000 | 36 | 2.24444 | | |

Selection Sort on accuracy. Cell means for these tests may be seen in Table 5.5. Analysis of variance was conducted with dependent variable accuracy. ANOVA results appear in Tables 5.6 and 5.7.

## 5.8 Discussion

The first result observed was the effect of labeling on the Quick Sort Posttest. Those in the label condition required a slightly longer time on the Posttest. ( $F = 3.07$, d.f. 1,39, $p = 0.08$).

There was no significant result relating to accuracy on either post test. This would

Table 5.7: Accuracy on Selection Sort Post-test

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| MEAN | 2175.62500 | 1 | 2175.62500 | 920.36 | 0.0000 |
| LABEL | 0.22500 | 1 | 0.22500 | 0.10 | 0.7595 |
| FIRST ALG | 0.02500 | 1 | 0.02500 | 0.01 | 0.9187 |
| LA | 2.02500 | 1 | 2.02500 | 0.86 | 0.3608 |
| ERROR | 85.10000 | 36 | 2.36389 | | |

Table 5.8: Time Results Selection Sort, Based on First Algorithm

| Cell Means | | |
|---|---|---|
| TIME | Quick Sort First | Selection Sort First |
| Labeled | 302.4 | 416.3 |
| No Labels | 335.5 | 393.9 |

indicate that labels on the data elements are not a major factor in understanding of the algorithm, even though students request them. In light of these results, it would seem that incorporating labels on data elements into animations is not a significant help. However, since the labels do not impact in a negative fashion on test accuracy and since students desire them, they may well be used in the animations when the representation allows it.

Another result was the effect of the first algorithm seen on the time for Selection Sort Posttest. Those who saw the Selection Sort first took longer on the Selection Sort post-test. These results appear in Table 5.8. Apparently, having learned the system and the representation on the Quick Sort was a positive influence on the simpler Selection Sort when it was second. Previous experience aids learning.

It is difficult to draw any definitive conclusion for animation design from this experiment. In the case of Quick Sort, the labeled condition took longer. However, after consideration, speed is not the best operationalization of understanding. Observation by the experimenter suggested that those who worked out answers rather than guessed took longer. People who had no understanding (as indicated by their comments) guessed at random. If this is true, than taking longer for the labeled implies that the labeled condition is better, contrary to superficial expectations.

This experiment indicates that labeling as a design issue did not increase performance. This was true for both levels of task complexity. In view of the lack of significant effect on accuracy of either, it was decided to use the labels when possible, since students preferred labels. The next step in exploring the issue of animation design was to consider the

question of whether the animations should be designed to be passive in nature, allowing the creation of preplanned data sets, or interactive in nature, allowing students to enter their own data sets in response to individual questions or desires. The next chapter describes this experiment and its results.

# CHAPTER VI

# EXPLORING INTERACTIVE ANIMATIONS

## 6.1 Introduction

We continued to explore the question of designing animations. Another design issue which we addressed was the question of interactive use of the animations. We also addressed the question of task by including algorithms from two different domains in this experiment. The domains used were sorting and graph algorithms. This chapter presents an experiment based on use of the animations in teaching algorithms in a laboratory setting. The area of concern is whether it is more effective for the student to observe a series of prepared examples of the animated algorithm or to create his/her own data sets and observe the working of the algorithm on these data sets. Early surveys of students (See Chapter 4) indicated that students would like to be able to control the animation in ways such as the content of the data sets, the number of times the animation is seen, and the speed of the animation. Prepared data sets had the advantage of using the size ranges suggested by previous experiments as well as including examples of problem or unusual data sets. Student created examples had the advantage of close engagement of student attention in the learning process.

One of the factors selected as a focus for this series of empirical studies was the issue of student involvement in the use of the animations. During the early experimental studies, described in Chapter 4, students were asked whether they desired control of the number of times the animation was run, the data sets input, and the speed of the animation. In general, students desired to take a more active part in using the algorithm animations. These studies indicated that students would like to be in control of the number of times they view the algorithm animations as well as the data sets used.

The experiment under investigation dealt with the differences in learning demonstrated by students who were able to control the data set compared to those who viewed the algorithm with prepared data. Studies of computer aided learning have shown that increased involvement in the learning process by some focused activity leads to better performance on post tests. This study applies this principle to the area of animated algorithms.

In order to observe the effects of task on the use of the animations, an algorithm from the domain of graphs was included. This was Kruskal's Minimum Spanning Tree Algorithm (MST) which determines a series of edges which allow access to all graph nodes using a minimal sum of the value of edges. Students were given a written description of the algorithm, followed by a hands-on session where the algorithm animation was presented. One group of students watched as the algorithm was shown until they expressed readiness for the post test. The second group of students created their own data sets for the animation and were allowed to continue until they expressed readiness for the post test. All students were exposed in counter balanced order to two algorithms, Quick Sort and Kruskal's Minimum Spanning Tree.

Results indicated that students in the active session performed at a higher level, as measured by post test scores. A hypothesis derived from this is that creation of one's own data set increases the level of involvement in the learning process and thus results in a higher degree of learning.

## 6.2 Experiment: Does Interaction Help?

Question: How much influence does learner control of data have on student performance on a post-test?

**Hypothesis:** Interactive involvement of the learner with the animation algorithm package increases understanding and application of the algorithm.

Table 6.1: Design of Experiment

| | DESIGN ISSUE | |
|---|---|---|
| ALGORITHM | Interactive | Passive |
| Kruskal's MST | | |
| Quick Sort | | |

## 6.3 Subjects

Subjects were seventeen students at the Georgia Institute of Technology who were taking CS1410, the first programming course for Computer Science majors at the Institute. These students were volunteers who received class credit for their participation.

## 6.4 Design

This was a 2 x 2 (within 2) design illustrated in Table 6.1. One variable was Style, Interactive or Passive. The second variable was the order of viewing of the algorithm. The within variable was the algorithm, Kruskal's MST and Quick Sort.

Algorithms from two different domains of computer science were used, sorting and graph algorithms. This allowed exploration of what types of algorithms are best presented through animated algorithms. Order was randomized to prevent an order effect. In addition, it was possible that one order might be preferable to another. Perhaps a simple sorting algorithm might better prepare one for the more complex graph algorithm. Alternatively the more inherently two-dimensional graphical algorithm might better prepare one for the more abstractly represented sorting algorithm. All these factors influence how animations are incorporated into teaching sequences.

## 6.5 Materials

Animated versions of the two algorithms created with the XTango algorithm animation software were used. These animations were similar to those used in previous experiments. Changes in the Quick Sort animation were the result of subject suggestions as described in Chapter 4. Students asked for an indication of active elements and data ranges in the data set. Thus, the current pivot in the Quick Sort algorithm was represented by a color change. The portion of the data range under consideration was bracketed by a pair of arrows pointing in the direction of search. They terminated on the partition boundaries and grew towards each other to represent the modification of the upper and lower indices. Since no effect of either representation or labeling was observed in the previous experiments, students preferences were followed in designing a vertical bar labeled representation. An example of this animation appears in Figure 6.1. Data sets were chosen in the medium size indicated from previous results. The data elements themselves were ordered in such a way as to provide (1) even-sized partitions and (2) minimal pivot movement (3) and maximal pivot movement.

The Kruskal Minimum Spanning Tree presents the graph as a series of nodes represented as open squares. The name of the node is represented by a capital letter. Letters are assigned to nodes in the order of their creation. As the edges are drawn, the value of the edge appears at the upper right of the window, labeled by end points. It also appears at the center of the edge. When the entire graph has been entered, the column of edge values is sorted by value. As an edge is added to the Minimum Spanning Tree, its color changes from green to red both in the graph and in the list of edge values. In the Kruskal Minimum Spanning Tree, edges not yet considered were blue, edges under consideration were flashed (blinked), while edges chosen for inclusion in the Minumum Spanning Tree were shown as red. Students in each group were either allowed to generate their own example data sets or were presented with prepared examples. Examples were chosen to show representative behaviors including a fully connected graph, an unconnected graph, and a simple geometric figure.

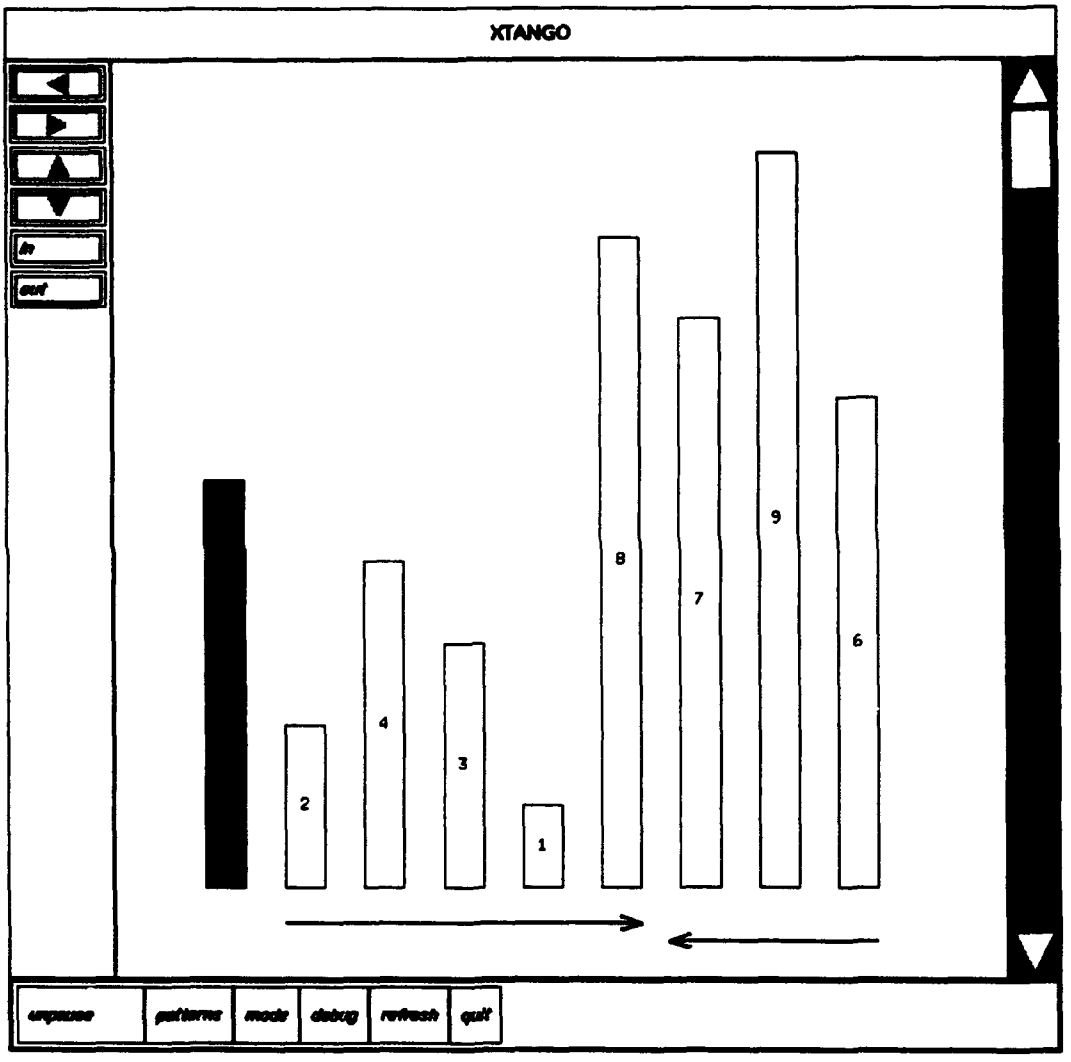



Figure 6.1: Revised Quick Sort Representation

Table 6.2: Number Correct of Thirteen, Kruskal MST

| | Kruskal First | Quick First | Marginal |
|---|---|---|---|
| Active | 9.60 | 9.25 | 9.43 |
| Passive | 8.00 | 5.00 | 6.50 |
| Marginal | 8.80 | 7.13 | 7.97 |

## 6.6 Procedure

Each student completed a pretest of computer knowledge and exposure. Student participants were exposed to two algorithms during the experimental session. Order was randomized to control an ordering factor, with half of the students seeing each algorithm first. The first algorithm was presented briefly through a paper handout, followed by the XTango animation. Students were allowed to interact with the animation or view prepared examples of the animation until they felt ready to complete the post-test. A post-test of understanding and application of the algorithm was given immediately following the demonstration. The animation was not available during the test. Then the subjects repeated the experiment with the second algorithm. Each student viewed both the Quick Sort, illustrated in Figure 6.1 and the Kruskal Minimum Spanning Tree, illustrated in Figure 6.2 algorithms.

## 6.7 Analysis

Dependent variables were accuracy and time on the post-test. 2 x 2 ANOVAs were computed. Factors used in the first analysis were interactive or passive condition and algorithm first seen.

Results (as shown in Table 6.2) indicated that those subjects who participated in an active session achieved higher scores on the Kruskal post-test than those subjects who were placed in a passive session.

The mean score for the Passive group were 50%, while the mean score for the Active

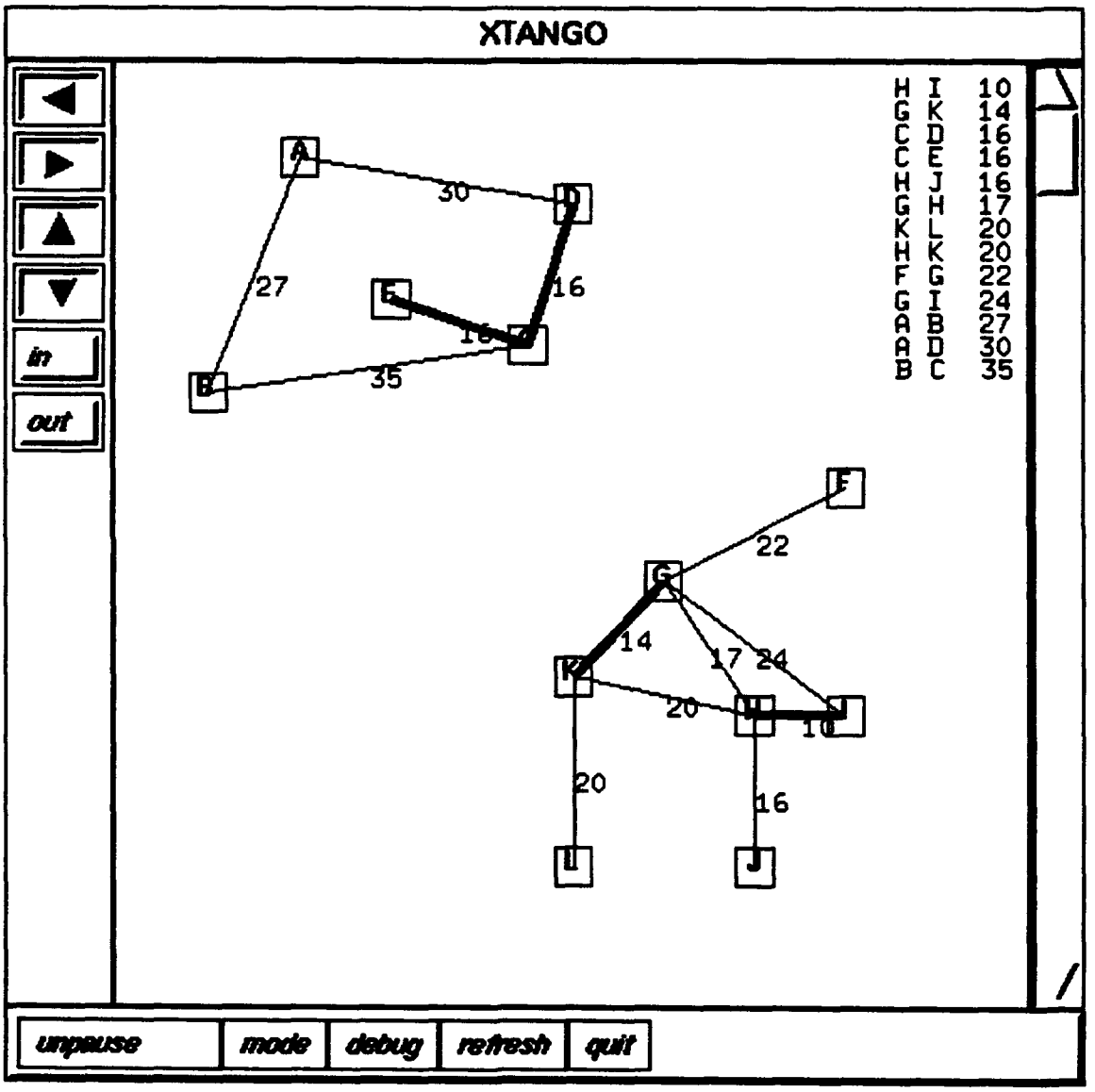



Figure 6.2: Kruskal's MST

Table 6.3: Accuracy Results, Kruskal MST

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 1067.81 | 1 | 1067.81 | 182.77 | 0.00 |
| FIRST ALG. | 11.81 | 1 | 11.81 | 2.02 | 0.18 |
| STYLE(Active/Passive) | 36.02 | 1 | 36.02 | 6.17 | 0.03 |
| AS | 7.39 | 1 | 7.39 | 1.27 | 0.28 |
| ERROR | 75.95 | 13 | 5.84 | | |

group was 73%. Analysis of variance showed this difference to be significant (F = 6.17, df= 1,13, p=.03) These results are indicated in Table 6.3. It is hypothesized that the greater level of involvement required in the interactive condition led to this significant difference.

Table 6.4: Accuracy Results, Quick Sort, Number Correct Out Of Twelve

| | Kruskal First | Quick Sort First | Marginal |
|---|---|---|---|
| Active | 4.40 | 4.75 | 4.57 |
| Passive | 4.75 | 4.75 | 4.75 |
| Marginal | 4.58 | 4.75 | |

Mean scores for Quick Sort appear in Table 6.4. No significant differences in accuracy were found for the scores on the Quick Sort post-tests. Table 6.5 summarizes these ANOVA results.

Analysis of the second dependent variable, time to complete the post-test, followed. No significant difference was noted in these results. ANOVA results are shown in Table 6.6 and 6.7.

It was noted, however, that groups with higher scores tended to take longer to

Table 6.5: Accuracy Results, Quick Sort

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 366.12895 | 1 | 366.12895 | 173.39 | 0.0000 |
| FIRST ALG. | 0.12895 | 1 | 0.12895 | 0.06 | 0.8087 |
| STYLE(Active/Passive) | 0.12895 | 1 | 0.12895 | 0.06 | 0.8087 |
| AS | 0.12895 | 1 | 0.12895 | 0.06 | 0.8087 |
| ERROR | 27.45000 | 13 | 2.11154 | | |

Table 6.6: Time Results, Quick Sort

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 7048350.02368 | 1 | 7048350.02368 | 41.92 | 0.0000 |
| FIRST ALG. | 82320.12895 | 1 | 82320.12895 | 0.49 | 0.4964 |
| STYLE(Active/Passive) | 177508.86579 | 1 | 177508.86579 | 1.06 | 0.3229 |
| AS | 3342.44474 | 1 | 3342.44474 | 0.02 | 0.8900 |
| ERROR | 2185859.45000 | 13 | 168143.03462 | | |

Table 6.7: Time Results, Kruskal MST

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 12495196.44474 | 1 | 12495196.44474 | 50.29 | 0.0000 |
| ALG | 82143.60263 | 1 | 82143.60263 | 0.33 | 0.5751 |
| STYLE | 289965.81316 | 1 | 289965.81316 | 1.17 | 0.2996 |
| AS | 167706.02368 | 1 | 167706.02368 | 0.68 | 0.4261 |
| ERROR | 3229877.95000 | 13 | 248452.15000 | | |

Table 6.8: Relative Results, Time and Accuracy

| | Kruskal Accur. 13 Questions | Kruskal Time In Seconds | Quick Accur. 12 Questions | Quick Time In Seconds |
|---|---|---|---|---|
| Active | 9.4 | 992.5 | 4.50 | 749.6 |
| Passive | 6.5 | 730.1 | 4.75 | 544.2 |
| Means | 7.9 | 811.3 | 4.67 | 647.3 |

complete the post-test. It was hypothesized that this was due to the fact that those who really understood the algorithm had to take time to work out the answers to the problems while those who had no understanding simply guessed an answer. The comparable results appear in Table 6.8.

## 6.8 Discussion

This experiment involved the comparative use of active creation of data sets and passive observation of prepared data sets in the Kruskal Minumum Spanning Tree and Quick Sort algorithms. This experiment has important implications for the use of animations in teaching algorithms. The groups of students who used the animation in an active way by entering their own data sets scored higher on the Kruskal test than did those students who only observed the animation. This indicates that involvement in the learning process is important for the effective use of the animation. This may be either an effect of heightened attention or the effect of being able to instantly create an example to answer any questions which arise. In either case, this form of participatory learning can have significant implications for the design of laboratories to accompany courses where algorithms are introduced. The availability of such animations could well serve to provide a learning tool which is readily adapted to individual student needs and concerns. Students could access the algorithm animations when they had a question about an algorithm or a series of animations in order to compare and contrast algorithms which are similar.

It is interesting to observe that effects were shown only on one of the two tasks involved. This suggests that animations may be best used in the teaching of certain tasks. One hypothesis is that a graph task may lend itself better to visual explanation than the numerically based sorting algorithm. Perhaps the student is better able to understand the mapping from the abstract domain of the algorithm to the visible domain of the animation for this algorithm. This would accord with suggestions made by Stasko, Badre, and Lewis [57].

This experiment helped to resolve one of the issues of design of animations, whether or not pre-prepared data sets should be used. User designed data sets are more effective in using the animation.

The next question to be addressed in the area of design is how to label steps in the algorithm. Two of the possible options are the use of text descriptions of steps of the algorithm and color cues to indicate the steps in the algorithm. These issues are explored in the next chapter.

# CHAPTER VII

# LABELING ALGORITHMIC STEPS

## 7.1 Introduction

In this chapter, we present an experiment which focused on two ways of labeling algorithmic steps. Labeling is done by means of color cues and by text explanations. Student demand for labeling of both types suggested this focus. In previous experiments, students using the animations were asked to indicate what features should be added to the animation in order to facilitate understanding of the algorithm. One suggestion was labeling of data elements. A previous experiment investigated the influence of labeling individual elements of data. This type of labeling was found to add little to understanding of the algorithmic process. Students also suggested that text should accompany the progress of the animation, to explain the process of the algorithm as it was visually displayed. This type of labeling is explored in this chapter.

A recurrent theme in the problem solving literature is that the more salient is the information derived from the example, the better are the problem solving results. An obvious method for increasing salience is to label or highlight important features. The influence of labeling in problem solving has a long history. One often cited example is Duncker's traditional candle problem box of thumbtacks and a candle are given to a problem solver. where the problem is more easily solved when the box of thumbtacks is labeled "box."

Much of the psychology based literature refers to labeling as the inclusion of textual names or descriptions, but Maguire[34] summarizes a body of research which stresses other types of cues such as highlighting, shape coding, color coding, and blinking.

The question arises as to whether certain features of information may be best presented by graphics or by text. Feiner and McKeown [22] suggest that rationale and abstract

information should be represented by text, while actions, location, and physical attributes should be represented by graphics.

Two possible types of information that could be included in the animation are the conceptual steps in the algorithm and the state of the data structure. Applying Feiner's suggestion to the design of animated computer algorithms would suggest that conceptual steps of an algorithm should be text labeled while actions taking place on the data structures would be represented in a graphical manner.

The experiment investigates whether either of these types of cues, a combination of both, or neither is most effective in teaching the steps of an algorithm in the animation. Four versions of Kruskal's Minimum Spanning Tree Algorithm were developed for this experiment. They included:

- Text/conceptual labels and visual color/action cues
- Text/conceptual labels but no visual color/action cues
- No text/conceptual labels but with visual color/action cues
- Neither text/conceptual labels nor visual color/action cues

A second focus of the experiment was whether there was a difference in versions in teaching concepts at a inferential level. It was determined to attempt measurement of this by the use of a transfer task. Transfer tasks are often used to test the ability of the learner to extract basic principles from the task or examples used in learning. A frequently cited example of this type of task is the problem of the General who must attack a city that may be entered by several roads, but is unable to send a large force along any road. This problem is transferred to a medical dilemma where an X-ray dosage large enough to destroy a tumor will also kill the patient if delivered to one spot.

Transfer or incidental learning may be detrimental, in that it may interfere with intentional learning. On the other hand, it may be helpful in that general principles are learned which may be applied to several related problem situations. With this in mind, it

was decided to add a transfer task to the experiment which would also use a greedy heuristic, since such a heuristic is at the heart of Kruskal's Minimum Spanning Tree Solution.

Results of the experiment indicated that textual labels indicating steps of the algorithm increased scores on the post-test for questions which required free responses, that is questions which required more conceptual application. The monochrome version of the algorithm proved more effective for the short answer and true/false questions. No significant difference was found in efficiency for the transfer task.

## 7.2 Experiment: Do Labels Help?

There are several ways to label steps in an algorithm. One method is to use visual cues such as color changes and codes to indicate actions. Another method is to display each step as a text description of what is taking place. These two methods are examined in the following experiment. The experiment seeks to determine whether text alone, visual cues alone, a combination of both, or neither is most effective in teaching the steps of an algorithm in the animations.

This experiment was conducted to answer two questions:

1. How much influence does the presence or absence of text descriptions of the algorithmic **conceptual** steps have on the effectiveness of animated algorithms in teaching computer algorithms?

2. How much influence does the presence or absence of color cues as indicators of algorithmic **action** steps have on the effectiveness of animated algorithms in teaching computer algorithms?

## 7.3 Subjects

Subjects were thirty-six students at the Georgia Institute of Technology. They were taking CS1410, the first programming course for Computer Science majors at the Institute. These students were volunteers who received class credit for their participation.

Table 7.1: Design of Experiment

| | CONCEPTUAL STEPS | |
|---|---|---|
| ACTION STEPS | Text Labeled | Unlabeled |
| Color | | |
| Monochrome | | |

## 7.4 Design of Experiment

This was a 2 x 2 between subject design. One variable was Text Labeled versus Non-labeled conceptual algorithmic steps. The second variable was Color Cued versus Monochrome Cued indicators of algorithmic action steps. This design is illustrated in Table 7.1.

## 7.5 Materials

Animations used for this experiment were created with the XTango Algorithm Animation Software. This software, developed by Dr. John Stasko of the Georgia Institute of Technology, allows the programmer to create graphical objects to represent data structures and to insert visual cues which represent the effects of the algorithm on the data structures. Figures 7.1 and 7.2 are samples of the algorithm.

Animated versions of the Kruskal's minimum spanning tree algorithm were used. Text labels reflected conceptual steps in the algorithm: namely,

- Select the next shortest edge.
- Check to see if the inclusion of the edge will cause a cycle
- If no cycle is created, add the edge to the MST.

Color was used in the following ways to represent algorithmic actions:

- Edges of the graph which had not yet been considered were blue.

- The edge of the graph under consideration was red.
- Edges selected for the Minimal Spanning Tree were black.

In addition, the type of line afforded redundancy in the actions of the algorithm by providing varied representations of the edges. All representations made use of thin, dotted, blinking, or thicker lines to represent untested edges, current edge under consideration, test for cycle, and inclusion in the minimum spanning tree, respectively. Thus, those subjects who were given both color and line cues received redundant coding of these actions steps of the algorithm. Those subjects who received both color and textual cues were actually given information in three ways, color, type of line, and accompanying text. A particular step in the algorithm might be shown in one, two, or three ways. The action might be shown by color change and type of line used while the rationale for the action appeared as a text message on the screen, thus providing three guides to the progress of the algorithm, based on two types of information. Of concern was the type of information provided which best led to understanding of the algorithm. The experiment sought to determine which combination of text and color cues was best for use in the animation.

### 7.5.1 Description of Algorithm

Subjects were given the following written description of the algorithm:

Kruskal's Minimum Spanning Tree Algorithm

A minimum spanning tree of a graph is an acyclic subset of the edges that connects all of the vertices and whose total weight is minimized. This algorithm illustrates a "greedy" strategy. At each step, the best possible choice of the moment is made.

The algorithm begins by sorting the edges by weight. Then the minimum spanning tree is built by adding edges.

Given an undirected graph G with set of vertices V and set of edges E, at each step add to the tree the edge of least weight which does not create a cycle. (A cycle is defined as a path of length at least one ($V_0$,$V_1$, ..., $V_k$) where $V_0 = V_k$. That is, in general, a cycle is

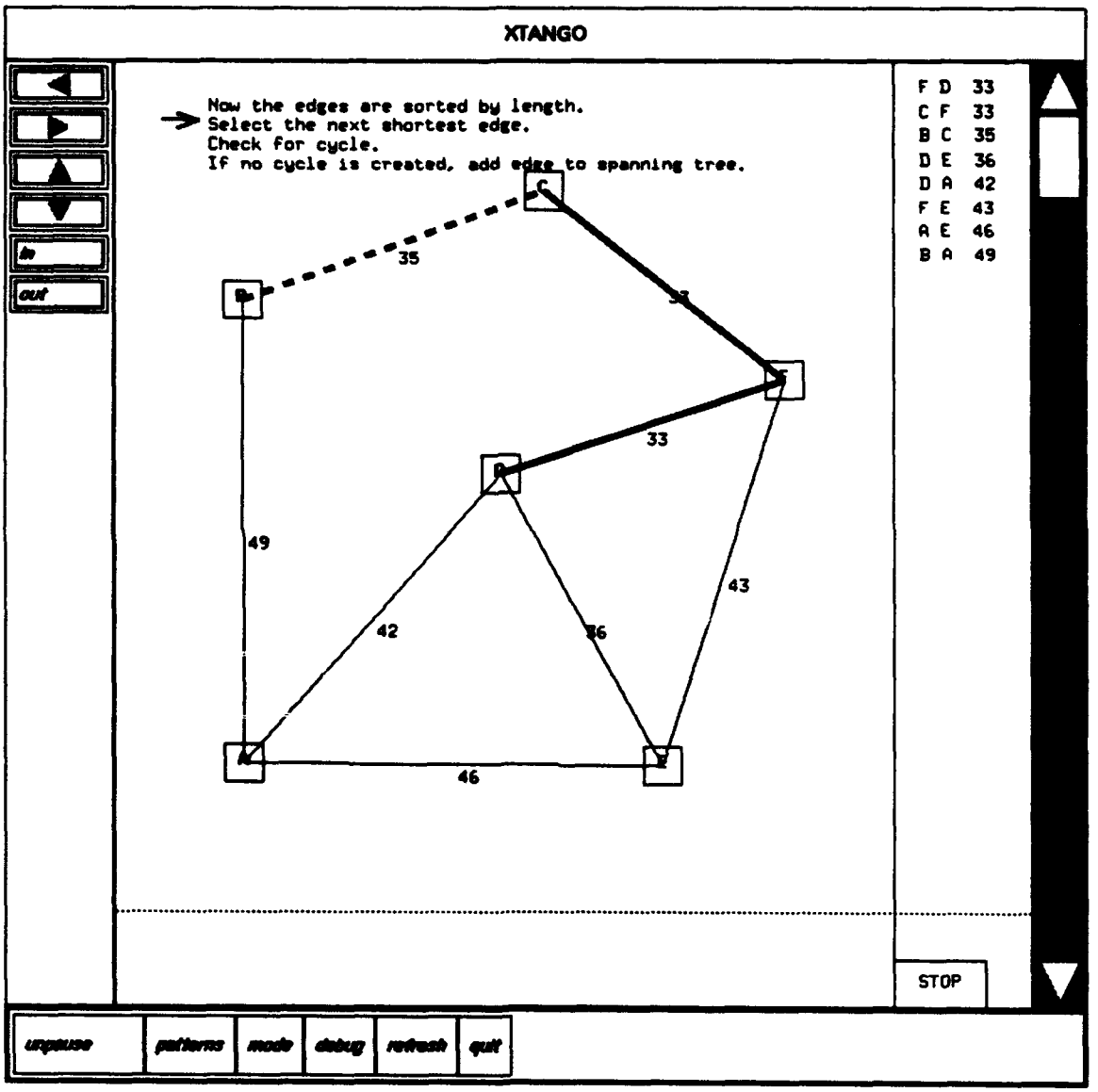



Figure 7.1: Revised Kruskal's MST with Text

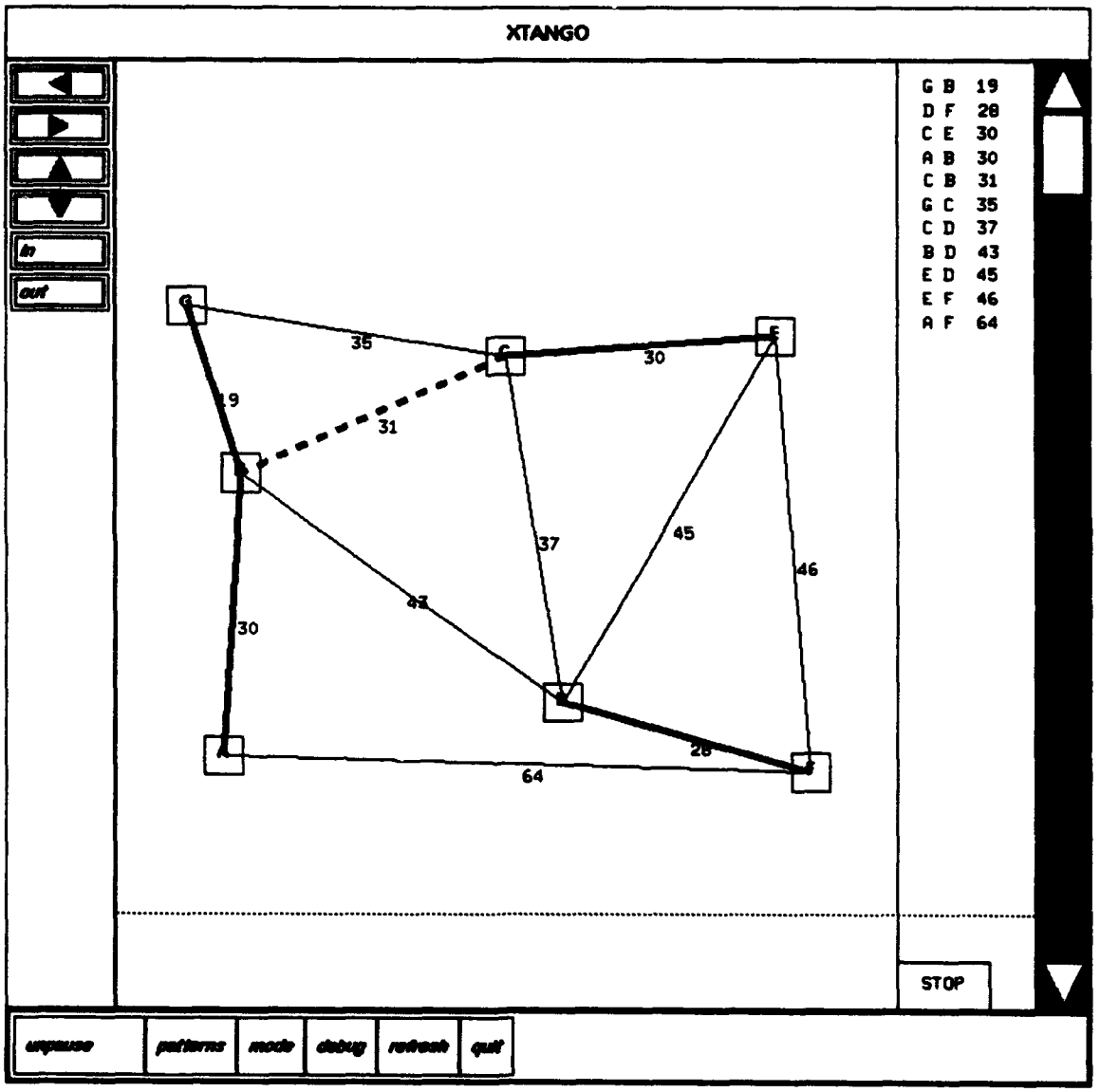



Figure 7.2: Revised Kruskal's MST without Text

a path of length three or more that connects a vertex to itself.) Continue until all vertices are examined. The minimum spanning tree will contain n-1 edges, where n in the number of vertices in the graph.

### 7.5.2 Post-tests

Three post-test instruments were developed. Learning objectives for the algorithm guided the development of these instruments. The objectives used were the following:

- The student will be able to list the steps in the algorithm.
- The student will be able to state the first step in the algorithm.
- The student will be able to apply the Greedy Heuristic to determine the next step at any point.
- The student will be able to recognize a cycle.
- The student will be able to determine when adding an edge to the MST creates a cycle.
- The student will not add an edge to the MST which creates a cycle.
- The student will be able to determine the advantages and disadvantages of this algorithm.
- The student will be able to apply the steps of the algorithm.
- The student will be able to identify the completion point of the algorithm.

The post-tests included:

- An on-line multiple choice and true/false questionnaire in which students applied knowledge about the algorithm to concrete examples. This test concentrated largely on the action portion of the algorithm. Questions were generally at the operational level.

- A paper short-answer questionnaire in which subjects were asked to generalize their knowledge of the algorithm. This test was more conceptually focused.
- A transfer task based on another greedy algorithm (the knapsack problem) in which both application and generalization questions were asked

Both post-tests of the minimum spanning tree algorithm were designed to address key concepts in the algorithm, from perspectives of both understanding and of application. These concepts were identified before the experiment, and questions were designed to approach these concepts through both recognition and description. The on-line questions were all of types that required only multiple choice or True/False response. However, in many cases one or several steps of the algorithm had to be carried out to determine which answer was correct. The paper post-test required the subject to

- Identify the portion of the algorithm which most contributed to its efficiency
- Recognize a key concept used in the algorithm
- Carry out one step of the algorithm
- Name a reason why a certain step could or could not be taken
- Carry out the complete algorithm

A transfer task was also developed. It was a series of questions based on the Knapsack Problem. This problem involves which several objects, each with a given weight and profit, must be placed in a knapsack of limited weight capacity to maximize the profit obtained. Heuristics often given for this problem are (1) minimum weight first, (2) maximum profit first, and (3) realative ratio of profit to weight.

## 7.6 Procedure

Each student completed a pretest of graph knowledge. A textual description of the algorithm was presented briefly at the beginning of a laboratory session. Students created a graph and

Table 7.2: Mean Test Results

| TEST RESULTS ACCURACY | | | | |
|---|---|---|---|---|
| | No Color No Text | No Color Text | Color No Text | Color Text |
| Pretest: 9 questions | 3.5 | 3.7 | 3.8 | 3.8 |
| On-line: 16 questions | 11.4 | 10.9 | 8.6 | 9.4 |
| Paper: 7 questions | 5.3 | 6 | 4.2 | 6.0 |
| Transfer: 5 questions | 1.2 | 1.2 | 1.6 | 0.9 |

observed the workings of the algorithm on that graph. Each student repeated the process of graph creation and running the animation until he or she felt ready to take the post-test, however each student created a minimum of three graphs. A post-test of understanding and application of the algorithm was given immediately following the demonstration.

## 7.7 Analysis

Table 7.2 contains the mean accuracy scores for each of the test instruments. Each test was analyzed separately with a 2 x 2 ANOVA. The pretest was checked to determine if the subject groups differed in prior knowledge of graphs. The on-line test measured behavioral objectives with fixed choice questions. The transfer test measured transfer to a conceptually similar problem. Of all the tests, only the on-line and paper tests had significant results.

For the on-line questionnaire, dependent variable accuracy, color was significant ($F = 6.31$, $df = 1,28$, $p = .018$) with higher accuracy for no-color. Anova results appear in Table 7.3.

The graph in Figure 7.3 demonstrates these results. This led to two hypotheses. First, the lack of color forced a higher degree of concentration in order to understand the algorithm, thus resulting in better storage of the concepts. Second, that the color distracted from the key concepts of the algorithm.

For the short answer paper test, text cues were significant ($F = 5.29$, d.f. $= 1,28$, $p<$

Table 7.3: Analysis of Accuracy Results for Kruskal MST, On-Line

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 3200.68245 | 1 | 3200.68245 | 550.39 | 0.0000 |
| COLOR | 36.71591 | 1 | 36.71591 | 6.31 | 0.0180 |
| TEXT | 0.18048 | 1 | 0.18048 | 0.03 | 0.8614 |
| CW | 3.54859 | 1 | 3.54859 | 0.61 | 0.4413 |
| 1 ERROR | 162.82937 | 28 | 5.81533 | | |

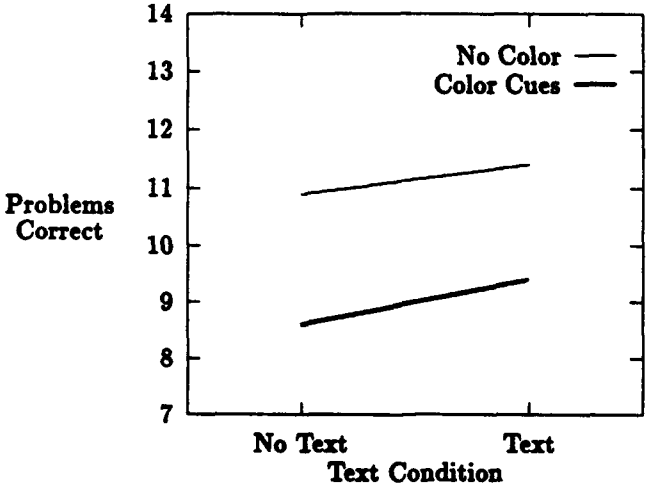



Figure 7.3: Results of On-line Post-Test

Table 7.4: Accuracy Results for Kruskal MST, Paper

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 914.85192 | 1 | 914.85192 | 382.01 | 0.0000 |
| COLOR | 2.09602 | 1 | 2.09602 | 0.88 | 0.3575 |
| TEXT | 12.67870 | 1 | 12.67870 | 5.29 | 0.0291 |
| CW | 2.09602 | 1 | 2.09602 | 0.88 | 0.3575 |
| 1 ERROR | 67.05556 | 28 | 2.39484 | | |

Table 7.5: Accuracy Results of the Pre-Test

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 438.57837 | 1 | 438.57837 | 385.45 | 0.0000 |
| COLOR | 0.38152 | 1 | 0.38152 | 0.34 | 0.5672 |
| WRITTEN | 0.19255 | 1 | 0.19255 | 0.17 | 0.6839 |
| CW | 0.02719 | 1 | 0.02719 | 0.02 | 0.8783 |
| ERROR | 31.85913 | 28 | 1.13783 | | |

0.03). These results are indicated in Figure 7.4. This result showed text cues with visual cues were better than visual cues alone. The textual cues helped to confirm the information indicated by the visual cues. Color had no significant effect on this particular result.

The combined results of the tests indicated that conceptual information which requires words to answer may be best given by textual means while the general actions or workings of the algorithm itself may be as readily conveyed through visual cues.

The pre-test and transfer task were also analyzed. The pre-test was designed to determine whether preexisting knowledge affected post-test scores. No significant difference was found among groups taking the test. The ANOVA results appear in Table 7.5.

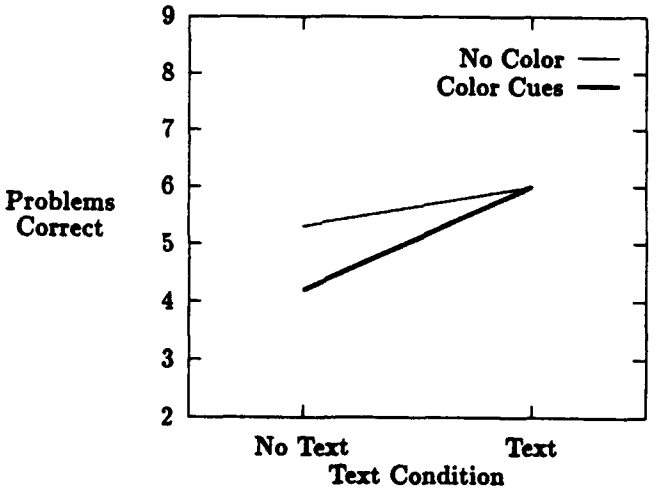



Figure 7.4: Results of Paper Post-Test

Table 7.6: Accuracy Results of the Transfer Task

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 337.11118 | 1 | 337.11118 | 224.21 | 0.0000 |
| COLOR | 2.14343 | 1 | 2.14343 | 1.43 | 0.2437 |
| WRITTEN | 0.70565 | 1 | 0.70565 | 0.47 | 0.4996 |
| CW | 0.70565 | 1 | 0.70565 | 0.47 | 0.4996 |
| ERROR | 37.58929 | 25 | 1.50357 | | |

Study of the transfer task results also indicated no significant differences among groups, ANOVA results for this task appear in Table 7.6.

## 7.8 Discussion

Results indicated a significant difference in two aspects. The first was that the color group has lower scores for the online test. This indicates that color, although popular with the users, may be a distractor. An alternative hypothesis is that lack of color requires more concentration to gain a feeling of understanding of the algorithm and therefore led to more memory cues being processed. This resulted in increased retention and understanding. However the groups with written steps had higher scores on the paper post-test, which were questions requiring free response answers concerning both concepts and application of the algorithm. This would seem to indicate that the optimal condition for these animations came with the uncolored version containing the written steps. This version ranked a close second on the online post-test and tied with color/words for highest score on the paper post-test.

These experimental results supported previous conclusions that graphics present actions well while conceptual steps may be best presented by text. As a result of this experiment, monochrome animations combined with textual cues were selected as the best

choice for conveying all types of information. Animations used in later experimentation were developed with these features.

This experiment was the last in this series of those which focused on design elements and task suitability. It provided interesting results which could be readily applied to the design of animated algorithms. At this point, we were ready to move on to the issue of how best to incorporate the animations into the teaching process.

Results of the student preference and design phases of the work led to the following design guidelines:

- Vertical bars with numeric labels are preferred for sets of numbers
- Data items should carry value labels where this is appropriate
- Tasks of an inherent spatial nature show more effect of the use of animations
- Monochrome animations are preferred
- Text explanations of algorithm steps should be included
- Animations should be designed to allow student interaction in selecting the data set

Using these guidelines it is possible to prepare an animation which can be used to accompany the teaching of computer algorithms. At this point, we began the next phase of the experimentation. This involved ways to use the animations in actual teaching of the algorithms. Several pilot studies were carried out to determine whether text or animation should come first in classroom presentation and to determine the optimal combination of animation, text, or both. The results of these studies led to a larger experiment which dealt with teaching the algorithms in small classroom groups to determine which approach is optimal for use of these animations as a course supplement.

# CHAPTER VIII

# EXPLORING LEARNING WITH ANIMATIONS

## 8.1 Introduction

The previous experiments were concerned with the issues of animation design. Previous experiments led to the hypothesis that data representation was not the crucial issue in designing the algorithm animations. They also indicated that students preferred labeled data with vertical bars representing data when it was appropriate. Other results indicated that interactive use of an animation was important, that color served more as a distractor than as a distinguisher, and that text explanations were helpful. Using the results of these issues to modify the design of the animations, we proceeded to the next stage on how the animations were to be used in teaching the algorithms.

At this point, the question was raised of exactly what part did the animation play in the learning process? Was the animation a help to textual or lecture presentations? Was the animation sufficient alone instead of being jointly used with textual presentations? Did the animation provide the basic concepts on the procedure of a particular algorithm, or did it instead serve to clear up problems, questions, and misconceptions after the foundation of conceptual understanding had been laid by a previous presentation?

These questions were explored in two ways. The first was an experiment in which the order of presentation of text and animation was studied. In this experiment, half the participants received the textual presentation first, the other half viewed the animation first. They received no other description of the algorithm than the name. This was designed to explore the question of which of the two, text or animation, was better for initial exposure to the concepts. No significant difference was found between the two groups used in this study.

A second exploration was a pilot study of discovery learning with the animations. In this study, groups of students were presented with an animation, with a text description of an algorithm, or text plus animation, and were asked to derive the rules of the algorithm this animation represented. This pilot study provided insight into how students derived information from the animations. The participants were able to derive many rules from the animation alone in simpler algorithms; they were not able to derive rules as readily from a more complex algorithm.

Students in the text only or text plus animation groups were also readily able to derive the rules of the simpler sorting algorithms which were presented to them. It was observed that the text added little to the conceptual formation and rule derivation of the algorithm. No significant difference was found in the post-tests.

## 8.2 Which Came First, the Text or the Animation?

This experiment explored the question of whether a text description served better as an initial presentation, followed by an animation, or whether the reverse was true and the animation was the better predecessor, followed by a text description of the algorithm. Since the senses are engaged in a different fashion when reading text than in watching a visual presentation of an algorithm, one method might be preferable to another in terms of the best effect on overall comprehension of the algorithmic concepts. This would be of interest in planning presentations of new algorithms in a classroom or laboratory setting.

### 8.2.1 Subjects

Subjects were twelve students enrolled in CS 1410, Introduction to Programming. These students were volunteers who received class credit for their participation in the experiment.

### 8.2.2 Design

This experiment was a 2 x 2 design in which subjects viewed two algorithms, Selection Sort and Kruskal's Minimum Spanning Tree Algorithm. Order of presentation was the first between factor and was randomized. The second between factor was whether students

Table 8.1: Design of Experiment

| Presentation Method | | |
|---|---|---|
| | Text First | Animation First |
| Selection Sort First | | |
| Kruskal MST First | | |

viewed the animation first or were given the text description first. The design appears in Table 8.1.

### 8.2.3 Materials

Two versions of XTango animations were prepared for this experiment. One was a labeled vertical bar representation of Selection Sort. Previous experiments indicated that this representation was preferred by students and that student performance with this representation was at least as good as any other. An animation of Kruskal's Minimum Spanning Tree was also prepared. Post-tests containing questions about the algorithms and their application were also prepared. These post-tests may be found in Appendix I and Appendix J.

### 8.2.4 Procedure

Assignment to the different conditions was random, with an equal number of students being assigned to each cell. The different conditions were:

- Text First, Kruskal First
- Text First, Selection Sort First
- Animation First, Kruskal First
- Animation First, Selection Sort First

Table 8.2: Number Correct of Ten Questions

| | Animation First | Text First | Marginal Means |
|---|---|---|---|
| Selection Sort | 7.3 | 8.7 | 8.0 |
| Kruskal MST | 4.0 | 4.3 | 4.2 |
| Marginal Means | 5.7 | 6.5 | |

The experimenter read a brief description of the experiment that was about to take place. Each subject then signed a consent form. Students were allowed to interact with the animation by entering data sets and observing the process of the animation upon these data sets. Students were also asked to give reasons for selecting the particular combinations of data sets that they used. After students completed both the animation and a reading of the text description, they completed an on-line questionnaire which covered concepts and applications of the algorithm. This process was then repeated with the second algorithm.

### 8.2.5 Analysis

Analysis of Variance was conducted with the data to determine if order of presentation (Text, Algorithm) or order of algorithm (Selection Sort First, Kruskal MST First), resulted in a significant difference in test scores. No significant difference was observed in accuracy for either Selection Sort or Kruskal.

It was observed, however, that those in the Text First condition averaged higher scores (65%) than those in the Animation First condition (57%) though not at significant levels. These scores appear in Table 8.2.

An interesting marginal interaction ($F = 3.79$, $df = 1,12$, $p < 0.09$) was observed for the Selection Sort test. The ANOVA results appear in Table 8.3. Those in the Text First condition who saw Kruskal first performed very well on the Selection Sort test. Those in

Table 8.3: Accuracy Results of Selection Sort

| ANALYSIS OF VARIANCE | | | | | |
|---|---|---|---|---|---|
| SOURCE | SUM OF SQUARES | D.F. | MEAN SQUARE | F | TAIL PROB. |
| MEAN | 768.00000 | 1 | 768.00000 | 242.53 | 0 .0000 |
| ORDER | 5.33333 | 1 | 5.33333 | 1.68 | 0 .2305 |
| ALGORITHM | 1.33333 | 1 | 1.33333 | 0.42 | 0 .5346 |
| OA | 12.00000 | 1 | 12.00000 | 3.79 | 0 .0874 |
| ERROR | 25.33333 | 8 | 3.16667 | | |

the Animation First group who saw the Selection Sort first also performed very well on the Selection Sort test. The condition with the lowest average scores for this post-test was the combination of Animation First and Kruskal First. One possible explanation for this is that the animation first group was expecting a similar algorithm to Kruskal since there had been no text definition. Therefore, they may not have made the best judgement in determining important features of the algorithm from the animation.

The indication here is that order of presentation is not the key issue in improving concept formation when using algorithm animations. Instead other issues must be studied to determine the most effective ways to use computer animations in teaching computer algorithms. A trend toward higher scores for the text first group may be an indicator that it is the preferred approach. Another factor to consider is that these were relatively simple algorithms. Therefore, it is hypothesized that either text or animation could serve as an adequate initial exposure to the algorithm. This may not be true for more complex algorithms.

### 8.2.6 Discussion

One focus of this study was to ascertain how students used the animations to answer questions about the process of the animation. In order to accomplish this, it was decided to explore their motivations for creating the particular examples that they used. Students

were asked to give the reasons for selecting the data sets they used. Some of the reasons for the Selection Sort were:

- A different type of example
- An ordered set
- A very large number in order to watch it progress through the array
- Reverse order
- Large range of numbers (from smallest to largest)
- Random
- Out of order and wide range
- Tried to ensure much swapping
- My age, birth date, and favorite number

Many of the reasons given map closely to typical examples. Frequently we discuss the process of a sorting algorithm on an ordered set and a completedly unordered set (reverse order). The students often gave random as a reason for the first trial or two, then progressed to more organized reasons for selecting the data elements used, such as these reasons given for Kruskal's MST:

- To see how the algorithm works
- To try to confuse the program
- To try to figure out the next move before the program
- To determine results on a geometric shape
- Well-spaced nodes
- To see if the algorithm carried it out the way I would

These reasons illustrate the desire to interact with the program and to try to come up with difficult examples. If examples were prepared for use in advance, they would probably include some random typical cases and some extreme cases, just as the student examples did. An additional advantage of the student prepared cases is that questions may be addressed as they arise with a data set tailored to the purpose.

## 8.3 Pilot Study on Discovery Learning

This study explored the process by which subjects were able to derive rules about algorithms from viewing various algorithm animations. Subjects were students in a beginning programming class at the Georgia Institute of Technology. Rules were chosen as the criteria because they provided discrete units of information which could be matched with subject input.

This study was designed to provide insight into exactly what part the animation played in understanding a new algorithm. Was the animation alone sufficient to provide all concepts necessary, or was it necessary to set the stage with a text explanation of what was to occur before or during exposure to the animation?

The question of concern here was whether animation alone, text alone, or animation plus text was the optimal means of presenting a new algorithm to a learner. This was a pilot study which was expected to provide guidelines for continuing the formal experimental sequence.

### 8.3.1 Design

There were three groups in this study. The first group was exposed to Animation Only. The second group received Text Only. The third group received both Animation and Text.

### 8.3.2 Subjects

The subjects were nine students enrolled in CS 1410, Computer Programming I. Students received class credit for volunteering to participate in the experiment. Subjects were engineering majors with classifications ranging from freshman to junior who were enrolled in

the computer science course as a cognate course for their engineering majors.

### 8.3.3 Materials

Animations of three sorting algorithms were used. Rules for these algorithms were determined *a priori* by consultation with several teachers of computer science. These algorithms and their respective rules included:

- **Selection Sort**

  1. Begin with the first element; it is the current smallest one. It is the one to use for comparison.
  2. Compare this element with each element in turn.
  3. If a smaller element is found, make that the one for comparison.
  4. Continue steps 2 and 3 to the end of the list.
  5. When the end of the list is reached, swap the current smallest with the first element of the list.
  6. Repeat with each succeeding element until all elements have been placed and the list is ordered.

- **Radix Sort**

  Radix Sort divides the $n$ input numbers into bins by using the digits, beginning with the least significant digit first.

  1. Begin at the beginning of the total list.
  2. Place each number in a list according to the last digit. Place numbers ending in one in sublist one, etc. Place each succeeding number after the numbers in that sublist.
  3. When all numbers are entered, rebuild the total list by joining the short lists in order, one, two, ...

4. Repeat steps 1, 2, and 3 for the second digit of each number and each succeeding digit until all digits are used.
5. When all digits are used, the list is sorted.

- **Quick Sort**

The Quick Sort procedure is as follows:

1. Select a pivot element.
2. Start a pointer moving from the left, searching for a value larger than or equal to the pivot.
3. Start a pointer moving from the right, searching for a value smaller than the pivot.
4. When these are located, swap the two elements.
5. Continue moving the pointers from left and right, searching for the larger/equal and smaller elements.
6. Repeat the swap process.
7. The location where the pointers meet is called the partition point. Swap the pivot with the last smaller element.
8. Divide the list into left partition, right partition, and pivot. Repeat the process on each partition piece until each piece contains only one element.
9. Combine the pieces to obtain the sorted list.

### 8.3.4 Procedure

Subjects were asked to read and sign a consent form. They also completed a demographic questionnaire to determine their computer science background.

Subjects in the Animation Only group were shown an animation of an algorithm. Subjects in the Animation/Text group were given a text description of the algorithms and were also allowed to view the animation. Subjects in the Text Only group were only allowed

to read text descriptions of the algorithm. Order of presentation was random. After the animation was completed, subjects were asked to write the rules they had derived from their observation. The experimenter read the generated document and compared it with the rules list previously described. This process was repeated and subjects were asked to add any features they had noted. This cycle was continued until no new algorithmic features were added.

After this plateau was reached, subjects were tested for their overall knowledge of the algorithm and ability to apply the process of the algorithm.

The process began again with a different algorithm.

After two algorithms, subjects were asked to apply the rules of the algorithms on a paper and pencil problem.

This aspect of the experiment was followed by an animation of the Quick Sort algorithm. This algorithm was not tested by a post-test. It served as a comparative measure of the ease of extracting rules from the presentation.

### 8.3.5 Analysis

Results of the experiment are given in the following tables. Table 8.4 contains the number of times the subject had seen the algorithm before he/she derived a particular rule for the Radix Sort. Table 8.5 contains the rule derivation for the Selection Sort.

The experimenter noted that after subjects had derived a plateau of rules, further viewings of the animation only produced details of the graphical implementation such as highlighting or blinking of elements. In one case, the only comment was the size of the data set.

### 8.3.6 Discussion

The discovery phase of the experiment led to several conclusions. One of these was that the Selection Sort algorithm was easily derived from the animation. Students generally were able to state all rules from no more than two showings. The Radix Sort was also easily conceptualized from the animation alone. One rule in the Radix Sort which was not so

Table 8.4: Viewing After Which Subjects Generated Radix Sort Rules

| RADIX SORT | | | | | | |
|---|---|---|---|---|---|---|
| Animation Only | | | | | | |
| | S1 | S2 | S3 | S4 | S5 | |
| Rules | | | | | | Underived |
| 1 | 1 | | 1 | | | 3 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 2 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 | 1 | 0 |
| 5 | 1 | 2 | 1 | 1 | 1 | 0 |
| | Animation Plus Text | | | | | |
| | S6 | S7 | | | | |
| Rules | | | | | | Underived |
| 1 | | 1 | | | | 1 |
| 2 | 1 | 1 | | | | 0 |
| 3 | 1 | 1 | | | | 0 |
| 4 | 1 | 1 | | | | 0 |
| 5 | 1 | 1 | | | | 0 |
| | Text Only | | | | | |
| Rules | S8 | S9 | | | | Underived |
| 1 | | | | | | 2 |
| 2 | 1 | 1 | | | | 0 |
| 3 | 1 | 1 | | | | 0 |
| 4 | 2 | 1 | | | | 0 |
| 5 | | 1 | | | | 1 |

readily derived was that the sublists must be maintained in numerical order for reassembly. In all cases, after two showings of the algorithm, a third showing brought no new algorithmic information. Some students phrased a rule in more detail after the third showing. In other cases, a further showing led to the student describing details of the graphical display. These graphical elements, which served as highlight or attention drawing devices to portions of the algorithm, were sometimes added to the description. Sample highlighting included swapping, comparing, and in-place.

Table 8.5: Round on Which Subjects Generated Selection Sort Rules

| SELECTION SORT | | | | | | |
|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | S5 | |
| Rules Animation Only | | | | | | Underived |
| 1 | 1 | 1 | 1 | 1 | | 1 |
| 2 | 1 | 1 | 1 | 1 | | 1 |
| 3 | 2 | 1 | | 1 | | 2 |
| 4 | 1 | 1 | 1 | 1 | | 1 |
| 5 | 1 | 3 | 1 | 1 | | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 0 |
| | Text Only | | Animation Plus Text | | | |
| | S6 | S7 | | | | Underived |
| Rules | | | | | | Underived |
| 1 | 1 | | | | | 1 |
| 2 | 1 | | | | | 1 |
| 3 | 1 | | | | | 1 |
| 4 | 1 | 1 | | | | 0 |
| 5 | 1 | 1 | | | | 0 |
| 6 | 1 | 1 | | | | 0 |
| | Text Only | | Animation Plus Text | | | |
| | S8 | S9 | | | | Underived |
| Rules | | | | | | Underived |
| 1 | 1 | | | | | 1 |
| 2 | 1 | | | | | 1 |
| 3 | 1 | | | | | 1 |
| 4 | 1 | 1 | | | | 0 |
| 5 | 1 | | | | | 1 |
| 6 | 2 | 1 | | | | 0 |

Table 8.6: Test Scores (Number Correct of 10) and Number of Algorithm Views

| Subject | Score on Radix Post Test | Times Viewed | Score on Selection Post-test | Times Viewed |
|---|---|---|---|---|
| Animation Only | | | | |
| 1 | 10 | 3 | 3 | 3 |
| 2 | 9 | 3 | 9 | 3 |
| 3 | 8 | 3 | 7 | 4 |
| 4 | 7 | 2 | 9 | 2 |
| 5 | 7 | 3 | 5 | 4 |
| Animation + Text | | | | |
| 1 | 9 | 3 | 9 | 2 |
| 2 | 8 | 2 | 9 | 2 |
| Text Only | | | | |
| 3 | 7 | 3 | 10 | 2 |
| 4 | 4 | 2 | 8 | 2 |

Another observation is that some parts of the algorithm appeared so obvious to students that they were not stated. An example is Rule One for Radix Sort: Begin at the beginning of the list.

The computerized test was followed by a paper and pencil test where each subject was asked to apply the two algorithms to a list of numbers. All subjects were able to correctly complete this task.

In contrast, students were also shown the Quick Sort Algorithm. This recursive algorithm is generally perceived as being more complex and more difficult both to teach and to learn. Students viewing this algorithm were less able to derive the rules involved. They were able to determine that the pivot changed and that comparisons were taking place, but some were not even able to determine the overall goal of the algorithm.

One subject noted that arrows moved in the direction of comparison and that comparisons were made with the marked pivot element. Exchanges were noted, but the motivation was unclear. In fact, the subject was unable to determine exactly what was happening and declared herself puzzled. Another subject decided the purpose of the algorithm was to

group single digit and double digit numbers. One subject, however, was able to derive the rules very precisely in two viewings.

Results were similar for all groups of studies, those with the animation only and those whose with either text or text plus animation. Although no formal statistical analysis was carried out due to the pilot nature of the study, it was noted that the group of students who received only the textual description scored lowest on Selection Sort (5.5 versus 8.5 and 7.4). (61% for Animation Only; 71% for Animation Plus Text; and 46% for the Text Only). This suggests that text alone may not be enough to convey the full meaning of this algorithm. Either the animation, repeated more than once, or text aided by the animation produced better post-test scores. However, this is a factor which may vary from algorithm to algorithm since all three groups scored very similar results in the Radix Sort (8.2,9,9).

It was noted that all groups found it difficult to modify the first description of the algorithm to obtain a more precise algorithmic description. Any modifications tended to be to the cosmetic description rather than to the procedural description.

#### 8.3.6.1 Conclusions

Observation of the subjects in this pilot experiment indicated that algorithms may be taught from an animation alone if

- The algorithm is simple.
- Algorithmic steps are emphasized.
- The subject is allowed more than one viewing.

The understanding of more complex algorithms, such as quick sort, is aided by the addition of algorithm animation. However, the animation alone is not sufficient to impart all the details of these more complicated procedures. Indications are that for the more complex procedures, the best approach is to combine verbal description with algorithm animation.

These experiments have investigated the coordination of text with the use of animation in teaching sorting algorithms. Results indicate that the combination of the two

Table 8.7: Comparison of Results for Conditions: Number Correct and Time in Seconds

| Subject | Selection | | Radix | |
|---|---|---|---|---|
| | 12 questions | | 10 questions | |
| Animation Only | | | | |
| 1 | 8 | 765.6 | 10 | 258.7 |
| 2 | 9 | 622.6 | 9 | 474.8 |
| 3 | 7 | 459.6 | 8 | 246.9 |
| 4 | 9 | 607.7 | 7 | 318.6 |
| 5 | 5 | 432.2 | 7 | 316.3 |
| Mean | 7.4 | 577.5 | 8.2 | 323.1 |
| Animation + Text | | | | |
| 6 | 9 | 591.5 | 9 | 253.9 |
| 7 | 8 | 668.8 | 9 | 341.8 |
| Mean | 8.5 | 620.2 | 9 | 300.3 |
| Text Only | | | | |
| 8 | 7.0 | 675.8 | 10 | 263.6 |
| 9 | 4.0 | 749.8 | 8 | 530.8 |
| Mean | 5.5 | 712.8 | 9 | 397.2 |

is better than either alone and that it is preferable to have the text presented first. It is interesting to observe that animation alone is sufficient for simple algorithms. It is the more complex algorithms which require that two learning methods be combined for maximum effectiveness. This would suggest that text and animation are better for separate and different aspects of concept determination and formation.

Our remaining studies, then, will continue to combine both a text presentation and an animated view of the algorithm. Text or a verbal lecture will be presented first, with the animation used to continue the conceptualization process by presenting an example of the algorithm just described.

# CHAPTER IX

# ANIMATION PRESENTATION TECHNIQUES

## 9.1 Introduction

This series of experiments has had two major goals, to design guidelines for presenting animated algorithms and to determine how best to use these animations in the teaching of computer algorithms. Previous experiments led to the following conclusions about the best format for presenting an algorithm though animation:

- Labels on data elements had little effect, either positive or negative.
- Textual labels accompanying algorithmic steps resulted in better results for questions requiring application of conceptual knowledge.
- Color cues were counter-productive in indicating steps of the algorithm.

In regard to the question of how best to use these animations, the previous experiment indicated that interactive data entry was better than passive observation. Students scored higher on the post-test when they were actively involved in learning by creating and entering their own example data sets as compared to those who observed examples that used previously prepared data sets.

These results were applied to animations for teaching algorithms in a a semi-classroom setting. The animations used included the textual labels of algorithmic steps. In addition, they were done using cues based on line-type and thickness rather than color. Several variables were examined:

- Type of presentation of the algorithm example to accompany the lecture, the two methods compared were animation or transparencies.

- Use of a laboratory session to clarify algorithm concepts.
- Student interaction with the animation during the laboratory session, where one group received prepared data sets and the second group used self-designed data sets.

Students participated in a group session. Varied conditions allowed some groups to participate in a lecture with an extra laboratory session while other groups participated in lecture sessions only. All lecture sessions were accompanied by an example of the algorithm which was either a series of transparencies prepared in advance or the same data set illustrated by an animation. Laboratory sessions were of two types, either using prepared data sets or allowing the student to create a series of personalized data sets.

Results indicated that there was no significant difference based on the type of example used with the lecture. Lecture plus laboratory groups performed significantly better than did lecture only groups. Further analysis indicated that the active laboratory sessions performed at higher levels than did the passive laboratory groups.

This experiment was an in-depth look at how animated algorithms may be used in the teaching of computer algorithms. In order to more closely match actual teaching of such algorithms, group sessions were used. One question which has arisen frequently in these studies is that of whether to use an animation or transparencies in a lecture presentation Another question is whether students will excel when given an additional laboratory session which allows them to observe several examples of the algorithm. A final question is the best format for the laboratory session, prepared examples designed by the lecturer or spontaneously generated examples designed by the student. In this case, the prepared data set examples can be planned to demonstrate various aspects of the algorithm. In contrast, students may be more involved when they create their own examples which may not include some of the unusual or problem aspects of the algorithm.

The question of how best to present material is an age-old question to the pedagogue. How, indeed, may one best transfer to others the concepts which are so clear to the teacher, yet such unknown territory to the learner? Felder and Silverman [23] among others, stress that students have many different ways of learning and that the learning and teaching styles

of both student and teacher affect the results of the teaching process. One aspect of this is the active/passive dimension which reflects whether students learn best by experimentation or by developing theories. This issue is addressed by the two laboratory formats used in the experiment.

## 9.2 Experiment on Teaching Style

Question: Does the use of an animation better present a complex algorithm than teacher drawn sketches or transparencies?

**Hypothesis:** An animated algorithmic example is better than static drawings for the understanding and application of the algorithm as measured by accuracy on a post-test.

Question: Is the lecture presentation of an algorithm enhanced by a laboratory session?

**Hypothesis:** Students given a laboratory session with the algorithm demonstrate a greater understanding of the algorithm as measured by performance on post-tests.

Question: What is the best format for a laboratory session using animated algorithms? Is student involvement preferable to prepared examples which can be crafted to present a wide range of possibilities and problems?

**Hypothesis:** Students who create their own algorithmic examples demonstrate a better understanding of the algorithm as measured by performance on post-tests.

## 9.3 Subjects

Subjects were students at the Georgia Institute of Technology enrolled in CS1410, the first programming course for Computer Science majors at the Institute. These students were volunteers who received class credit for their participation. Sixty-two students participated in the experiment.

Table 9.1: Design of Experiment

| Lecture Example | | | |
|---|---|---|---|
| LABORATORY CONDITION | | Animation | Prepared Slides |
| Lecture Only | | 15 | 15 |
| Lecture plus Lab | PassiveLab | 7 | 9 |
| | Active Lab | 7 | 9 |

## 9.4 Design

The design was a 2 x 2 (nested 2) design as represented in Table 9.1. One variable was presentation of the lecture example, using Animation or Prepared Slides. The second variable was Laboratory Session or Lecture Only. This design also encompassed a nested 2 level factor under laboratory session where the variable of concern was Laboratory Type, either Active or Passive.

The algorithm used was Kruskal's Minimum Spanning Tree Algorithm.

## 9.5 Materials

All students completed a consent form. They also completed a pre-test on knowledge of graphical concepts. These materials can be found in the appendix.

A lecture describing the algorithm was given to all groups. Students in the Lecture/Animation groups were stepped through an animated example of the Kruskal's Minimum Spanning Tree Algorithm created by the Polka Algorithm Animation software. Students in the Lecture/Slides group were shown the same example graph by means of a series of prepared transparencies. These transparencies were created from window-dumps of the Polka example.

The Polka animation package allows the creation of animations which are similar to the XTango animations but have additional features. The Polka animation package was selected for this demonstration since it allows step-by-step control as well as sequential

running of the animation. The example was prepared with the monochrome cues and textual algorithmic steps suggested by the results of the previous experiments. A sample frame from this package appears in Figure 9.1. For those students in the active and passive laboratory groups, a prepared sheet of instructions explained how to access the XTango animation of Kruskal's Minimum Spanning Tree Algorithm. The only difference in the two handouts was that the Active group was instructed how to create a graph while the Passive group was given the names of prepared data files.

The version of the animation used in this experiment was based on previous experiments which indicated that a monochrome version of the algorithm with algorithmic steps appearing as text was best as measured by performance on the post-test. This type of animation was used for both the lecture example and for the animation laboratory.

All groups completed an on-line test requiring application or understanding of the algorithm. This test appears in Appendix K. They also completed a free response test (contained in Appendix L) on paper which was designed to require that they put into words concepts relating to understanding the algorithm.

## 9.6 Procedure

Students were divided into 4 groups of approximately sixteen students each. The groups were Lecture/Animation, Lecture/Slides, Lecture and Lab/Animation, Lecture and Lab/Slides. Each laboratory section was divided into half Active and half Passive, providing sixteen students in each group. Students were randomly assigned to a particular sub-group. Each student completed a consent form and a pretest of graph knowledge. All students were exposed to a lecture presentation of the algorithm, accompanied by either the Polka animation or by the prepared slides. For those in the laboratory condition, the XTango animation followed. Students in the Active group created graphs and observed the workings of the algorithm on those graphs. Students in the Passive group were given a list of prepared file names and asked to observe the workings of the Kruskal MST Algorithm in the XTango environment on those files. All students were allowed twelve minutes for the laboratory

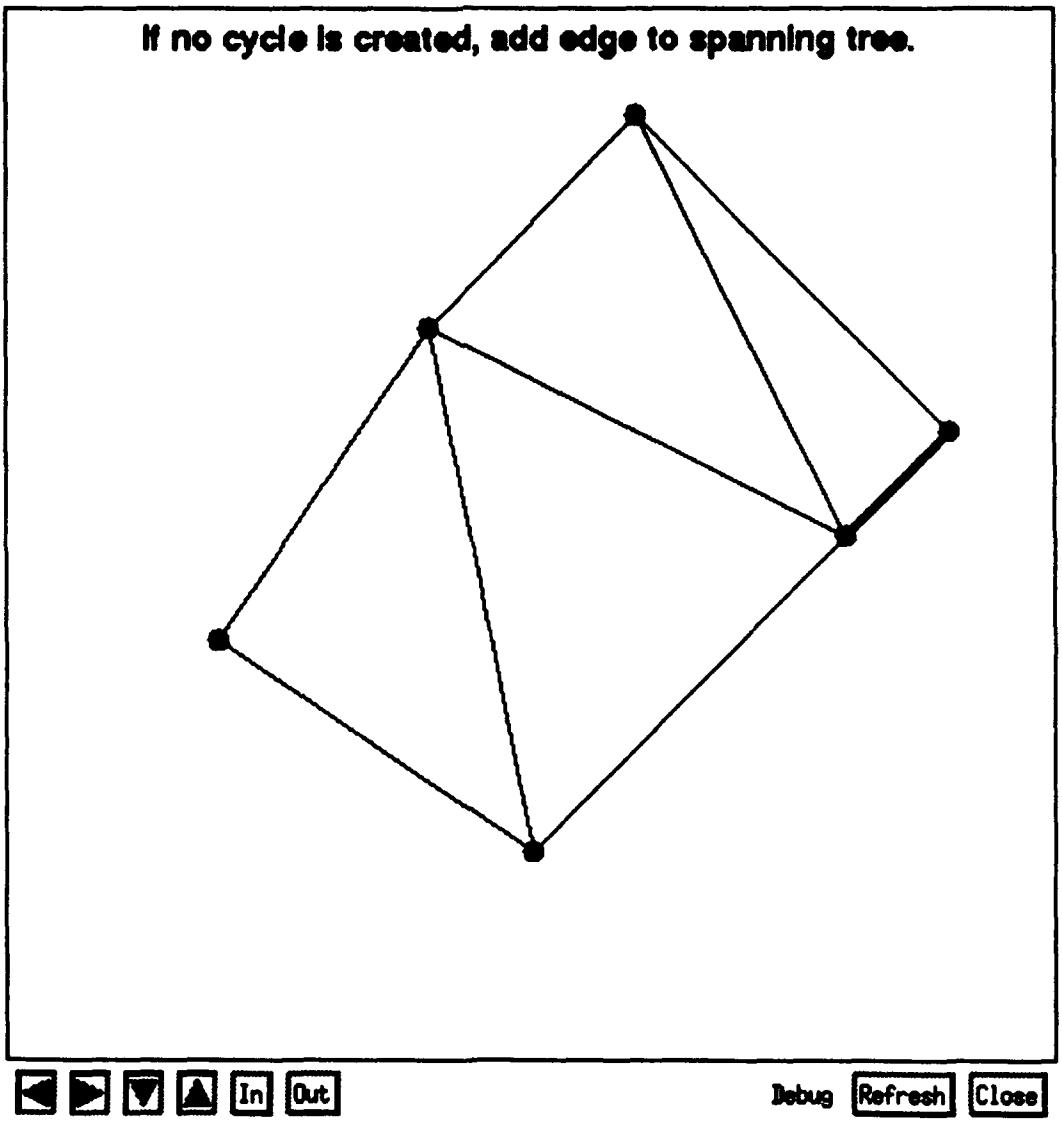



Figure 9.1: Kruskal's MST POLKA

session. The twelve minute time limit was derived from the previous experiment where it was determined that the average time a student spent experimenting with the graphs was twelve minutes.

Students then completed two post-tests on the algorithm, one of which was an on-line version composed of true/false and multiple choice questions. The second post-test required short answers, stating of concepts or explanations, and application of the entire algorithm.

## 9.7 Analysis

Accuracy, the dependent variable, was measured on two instruments, an on-line fixed choice test and a paper and pencil short answer test. This test was designed to measure behavioral objectives based on concepts and applications of the algorithm. Analysis of variance was used.

Analysis of the pre-test scores revealed no significant difference between groups of students participating in the experiment. A 2 x 2 ANOVA was conducted. The two factors in the first analysis were Lecture only versus Lecture/Lab and Polka Animation versus Prepared Slides. The factor used in the second analysis was type of laboratory session, Active or Passive.

Inspection of cell means for both the on-line and the free response test, which appear in Table 9.2 and 9.3, indicated that the active and passive laboratory groups scored higher than the no laboratory group. The active laboratory group had the highest scores on the free-response test. Statistical analysis was then undertaken in order to determine which of these differences were significant.

### 9.7.1 On-line test

The following results are based on a maximum possible of nineteen points. The questions on the on-line test were either true/false or multiple-choice in format. This format allows two techniques, recognition and guessing, which are not easily applied in free-answer style questions.

Table 9.2: Cell Means, On-line Test, Number Correct of Nineteen

| | | Lecture Example | Lecture Example |
|---|---|---|---|
| LABORATORY CONDITION | | Polka Animation | Prepared Slides |
| Lecture Only | | 11.87 | 11.8 |
| Lecture plus Lab | Passive Lab | 13.71 | 13.22 |
| | Active Lab | 13.83 | 13.89 |

Table 9.3: Cell Means, Free Response Test, Number Correct of Twenty-One

| LECTURE EXAMPLE | | | |
|---|---|---|---|
| LABORATORY CONDITION | | Polka Animation | Prepared Slides |
| Lecture Only | | 14.47 | 16.13 |
| Lecture plus Lab | Passive Lab | 16.43 | 16.67 |
| | Active Lab | 18.14 | 17.89 |

Table 9.4: Cell Means, On-line Test: Lab/No-Lab, Number Correct of Nineteen

| TEST RESULTS | |
|---|---|
| No Lab | 11.83 |
| Animation Lab | 13.45 |

Table 9.5: ANOVA On-line Test, Lab Condition

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| Mean | 9747.14 | 1 | 9747.14 | 684.75 | 0.000 |
| Lab/No-Lab | 39.93 | 1 | 39.93 | 2.80 | 0.099 |
| Error | 839.84 | 59 | 14.23 | | |

There was no significant difference between the two groups which have lectures accompanied by slides illustrating an example of the algorithm or lectures accompanied by an animation example of the algorithm. This may be explained by the fact that both groups were able to use visual techniques to supplement the algorithm and that either of these methods was adequate for the purpose.

In a comparison of groups which received the laboratory session, results indicated that students completing a laboratory session performed marginally better on the post-test ($F=2.80$, d.f.1,59, $p< 0.1$) as measured by the on-line test.

### 9.7.2 Free Response Test

The following results are for the paper post-test requiring statement or application of concepts. All questions are free-response in form. There were seven questions, each counted as three points, for a maximum high score possible of twenty-one points. Questions were

Table 9.6: Cell Means, Free response Test, Number Correct of Twenty-One

| Cell Means | | |
|---|---|---|
| | Slides | Animation |
| No-Lab | 16.1 | 14.5 |
| Lab | 17.3 | 17.3 |

Table 9.7: ANOVA Free-Form Test, Two-Factor

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Source | Sum of Squares | D.F. | Mean Square | F | Tail Prob. |
| Mean | 16311.93 | 1 | 16311.93 | 1179.76 | 0.00 |
| Lab/No-Lab | 60.35 | 1 | 60.35 | 4.36 | 0.04 |
| Animation/Slides | 10.57 | 1 | 10.57 | 0.76 | 0.39 |
| Interaction | 10.77 | 1 | 10.77 | 0.78 | 0.38 |
| Error | 801.93 | 58 | 13.83 | | |

designed to address basic concepts necessary for understanding of the algorithm in addition to a complete demonstration of the working of the algorithm on a provided graph.

Results of this analysis indicated that students who completed a laboratory session performed significantly better on the free-response post-test ($F=4.36$, d.f. 1,58, $p< 0.05$) than those who did not. The amount of difference in this result indicates that student laboratory participation is more effective for questions which require more conceptual knowledge than questions which require recognition of the individual steps of the algorithm.

### 9.7.3 Laboratory Style–Active, Passive

These results led to further study of the differences among the conditions based upon the type of laboratory session – active, passive, or none. Cell means indicated that those students in the active condition had the highest scores on the free-response test. These

Table 9.8: Cell Means for Three Lab Conditions, Number Correct of Twenty-One

| Cell Means | |
|---|---|
| No Laboratory | 15.3 |
| Passive | 16.6 |
| Active | 18.0 |

Table 9.9: Results, Free-Form Test

| ANOVA RESULTS | | | | | |
|---|---|---|---|---|---|
| Means | 15702.75099 | 1 | 15702.75099 | 1149.12 | 0.0000 |
| Lab Type | 77.31089 | 2 | 38.65544 | 2.83 | 0.0671 |
| Error | 806.23750 | 59 | 13.66504 | | |

results appear in Table 9.8. Analysis of Variance for the three possible lab conditions indicated that laboratory condition was significant ($F = 2.83$, d.f. 2,59, $p < 0.07$), as shown in Table 9.9. Pairwise t-test were performed to determine where the difference in condition actually lay. The significant difference ($p = 0.05$) was discovered between the active and the no laboratory condition.

## 9.8 Discussion

This experiment was interesting in several aspects. First, was the fact that the example used did not make a significant difference in teaching the algorithm. This seems to suggest that even though it is valuable to have a visual aid to concept formation, the animation, while enjoyable to the student, does not provide added clarity over some other visual representations. A second aspect of interest was that the advantage of the interactive laboratory session was confirmed. As before, it was found that these students excelled when compared to those in the passive laboratory condition as well as when compared to those students

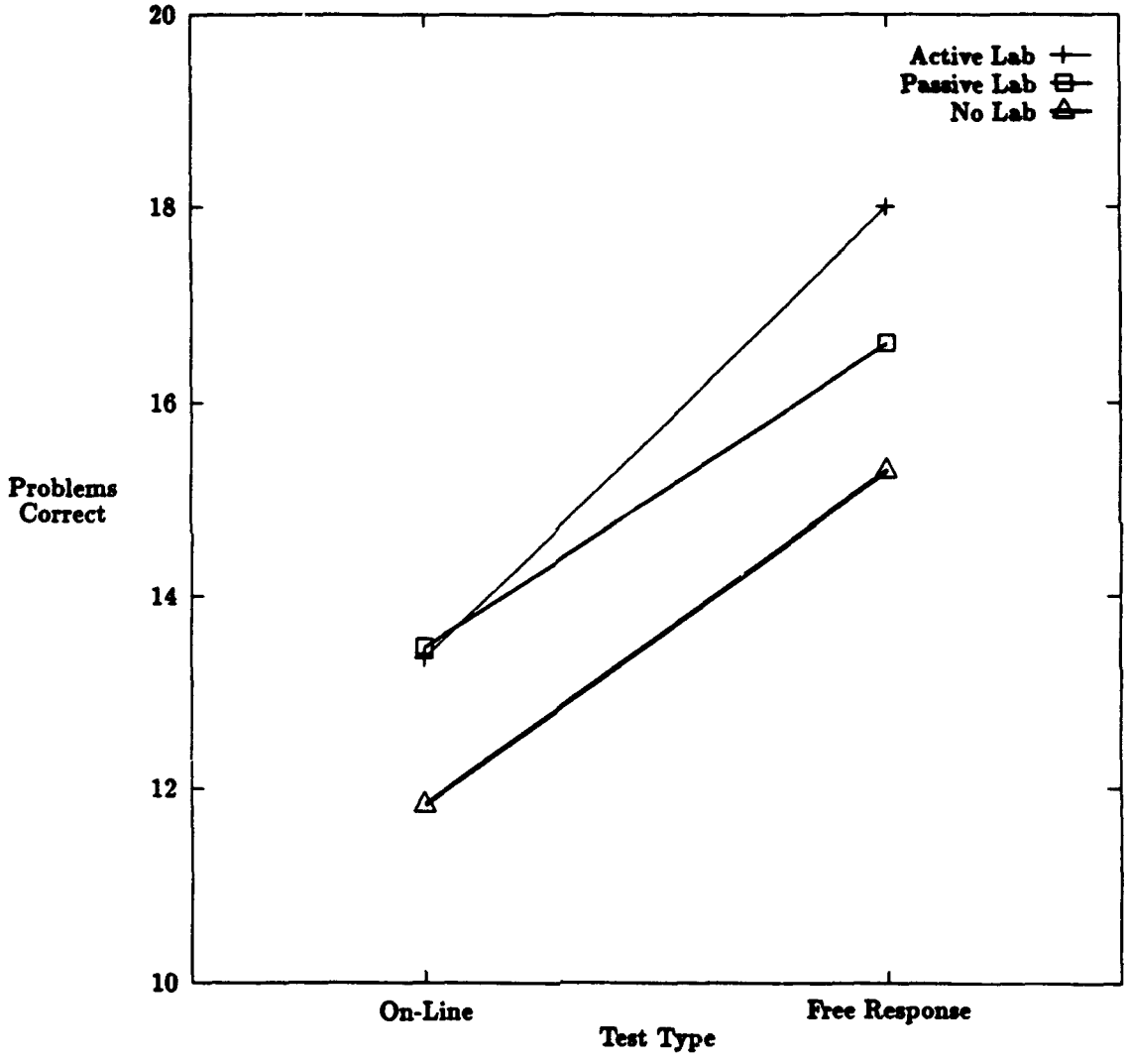



Figure 9.2: Cell Means, Three Lab Conditions

who did not participate in a laboratory session. Of special interest is the fact that the intuitive advantage of the laboratory group was not statistically supported. Simply having a laboratory session is not enough to improve performance; the issue of control and interaction must also appear. This is a strong indication that one valuable use of these animations is to make them available to the students outside the classroom setting. This might be done in either a closed laboratory or open laboratory setting where students would create sample data sets and observe the workings of the algorithms to be learned on these sample data sets. This is of special concern today when many computer science curricula are being revised to conform to the 1991 ACM guidelines. These guidelines encourage the inclusion of more closed laboratories in the beginning courses, but the design of these closed laboratories is not always clear to the curriculum developer. This result suggests that student active participation is a key issue in this design process.

A third feature of interest was that while those in the active laboratory performed at a slightly higher level for both portions of the test, the difference was larger for the free response test than for the on-line test. The nature of the two tests is important in understanding this result. In general, the questions on the on-line test required recognition of the correct response rather than generation of a response. These questions might be described as being more on a procedural and operational level than on a conceptual level. This speaks to the issue of what types of learning are most affected by the use of the animations. A previous hypothesis was that these animations may aid in concept formation. The results support that hypothesis.

### 9.8.1 Conclusions

The results of this experiment indicate that an advantage was shown for students receiving the XTango animated algorithm laboratory session. This advantage was more marked for those questions which required knowledge at a deeper level (the free-response test). Questions on this test required drawing conclusions from the questions asked as well as demonstrating a holistic version of the algorithm. Students who received the laboratory

session also performed better on the on-line true/false or multiple choice questions, but not at significant levels. This finding indicates that the animation session is more crucial for those conceptual questions than for the more basic operational level of question.

However, study of the results for the two laboratory conditions indicated that students who are in the active condition and create their own data sets for the algorithm achieve higher scores than those who observe prepared data sets. This results suggest that a study on the effect of student involvement in the learning process should be conducted.

However, no significant difference was observed for the animated algorithm over a more traditional approach of prepared transparencies for example used with the lecture.

# CHAPTER X

# CONCLUSION

## 10.1 Introduction

This series of experiments has covered several aspects of the use of animated algorithms in teaching computer algorithms. The initial points of study were student preferences and relationship to student performance. The results were applied to questions of design of the animations and appropriate tasks for their use. Finally these animations were used in actual teaching situations.

The first studies constructed the ideal format of an animation from a student learner perspective. Students were given the opportunity to compare and rate varied data representations used in animations of sorting algorithms. They were also asked to give suggestions for improving the animations. The student's suggestions focused on the representation of the data (vertical bars when appropriate, labeled data values) and the representation of the steps of the algorithm (major changes marked by text or by visual cues). The final study in this preference series was based on the question of the ideal representation of data. It served to measure whether preferences matched performance. This experiment considered factors of representation and data set size. A moderate or medium sized data set was indicated by the experimental results. However, no clear indication emerged that representation is the key issue in design of these animations. These studies constituted a foundation for the development of a second set of experiments which sought to resolve design issues. The second set also checked if these students perceived ideal animation was also the animation which led to fastest or most accurate learning.

The second set of studies dealt with design issues including representing data elements, labeling of data elements and student control of data sets, labeling of algorithmic

actions by color cues, and including of textual explanations explaining steps of the algorithm. The experiments allowed the development of guidelines as to which issues did, indeed, affect performance. The first experiment in this series focused on the issue of labeling data elements. Students were presented with animations which differed in the whether or not the data elements were numerically labeled and in the representation of the data elements. An experiment was also conducted which focused on the design issue of whether students should prepare their own experimental data sets or use prepared data sets. The third design experiment concerned two issues: (1) color compared to monochrome cues for algorithmic actions in the algorithm, and (2) the presence or absence of textual explanatory cues for conceptual steps of the algorithm. The third series of studies dealt with the question of exactly what role the animations played in the learning process. Was it better to precede the text by an animation? Should the animation come first instead? Secondary issues included such questions as was the animation alone sufficient for learning? Was text required as a help to building concepts through visual demonstration? What did the visual representation add in addition to a new and interesting display? Results of these studies indicated that the combination of text and animation was optimal, and that order of presentation (text first, animation first) did not matter. Also of importance was the discovery that simple algorithms may be taught from any of the three approaches – text, animation, text plus animation – with relative success. However, teaching more complex algorithms seems to require a multi-approach strategy where lecture or text is accompanied by an animation for maximum results.

These experiments raised the question of exactly how is the animation to be used in teaching. One possibility is for the animation to accompany a classroom lecture instead of transparencies or sketches. Another option is to provide the animation in a laboratory setting following a classroom presentation. With lab use of the animations, subjects were either given prepared data sets or required to actively create their own data sets. The experiment supported the use of the animation in an active laboratory setting.

## 10.2 Results

This research examined several aspects of animation design. Results from this set of experiments provide guidance to the design and use of animated algorithms. The experimental results are summarized in Table 10.1.

### 10.2.1 Student Preference

The first section of this research dealt with determination of student preferences and their correlation to performance. Student preferences were elicited in three ways:

- One group of students was shown three data representations of Quick Sort: vertical bars, horizontal bars, and dots. They were asked to rate these in a comparative manner.
- Twenty-one views of a sorting algorithm were created. These views differed by fill-style, labeling, and representation. Students ranked the views in order of most liked.
- Subjects were taught Quick Sort by reading a hand-out and viewing an animation. The animations varied in representation and data set size. Students were asked to give suggestions for changes in design of the animations.

The results were that students preferred vertical bars to horizontal bars and both of these to dots. They also desired numeric labeling of data elements and textual and visual cues to indicate steps and actions of the algorithm. No statistical advantage of accuracy was observed for any representation of the data in the sorting algorithms, so it was decided to follow student preferences in design by using vertical bars for these applications. Medium data sets were selected for use. These suggestions were also used to help guide the choice of which design issues should be investigated. Three of these suggestions became the focus of experiments: data labels, color emphasis of algorithmic points, and the addition of textual explanations.

## Summary of Results

Table 10.1: Summary of Experimental Results

| Focus | Algorithm | Subjects | Results |
|---|---|---|---|
| *Student Preference* | | | |
| Shape of Data Elements | Quick Sort | 32 | Prefer bars |
| Student Preference<br>Fill, Shape, Labels | Quick Sort | 25 | Prefer vertical bars<br>Request labels |
| Shape and<br>Data Set Size | Quick Sort | 36 | No signifcant difference<br>Students prefer vertical |
| *Design Issues* | | | |
| Labeled Data Elements | Selection Sort<br>Quick Sort | 40 | No accuracy effect of data labels<br>Keep labels to suit preference |
| Data Set Control | Kruskal MST<br>Quick Sort | 17 | Active control group<br>better accuracy<br>Kruskal $p < .04$ |
| Color labels and Text | Kruskal MST | 36 | Color detracts (online)<br>$p < 0.02$<br>Text steps help (paper)<br>$p < 0.03$ |
| *Teaching Use* | | | |
| Algorithm or Text First | Kruskal MST<br>Selection Sort | 12 | No significant difference<br>Trend indicates text first |
| Discovery Learning | Selection Sort<br>Radix Sort | 9 | Trend for text plus animation |
| Lecture Example Type<br>Addition of Lab<br><br><br><br><br><br>Lecture Example | Kruskal MST | 62 | Lab is better<br>(free-response)<br>$p < 0.05$<br>Active Lab is best<br>$p<0.05$<br>Marginally better with lab<br>(fixed response)<br>$p< 0.1$<br>No Significant Difference |

### 10.2.2 Design Issues

The next section of the research was to explore issues of design. The independent variables were related to objective measures of student performance. A secondary focus of this portion of the research was to consider suitable tasks and algorithms for the animations. The tasks were sorting and finding the minimum spanning tree of a graph. Algorithms used included Quick Sort, Radix Sort, Selection Sort, and Kruskal's Minimum Spanning Tree Algorithm. Three experiments were run.

The first experiment involved labeled and unlabeled data elements and two sorting algorithms. Students were presented with animations of both algorithms, Quick Sort and Selection Sort. Labeling was not found to have a significant effect on post-test accuracy. Because no effect of labeling was found, it was decided to continue to follow student preference in using labeled data elements.

The second issue of design was whether data sets should be user designed or prepared in advance by the animator or programmer. Subjects viewed animations of Kruskal's MST and Quick Sort. Results indicated that students who were allowed to create their own data sets achieved higher post-test scores for the Kruskal Minimum Spanning Tree algorithm. Possibly the greater effects of this algorithm (as compared to the Quick Sort algorithm) are due to a better natural visual mapping by the student of the graph algorithm to the animation. The result of this experiment was to include the active creation of examples by the user in the design process.

The last experiment in this series was a further study of the labeling issue. Although students had suggested a need for labels, labels of *data elements* may not be the key to algorithm understanding with an animation. Perhaps the items to be labeled are not the data items, but instead the steps of the algorithm itself. This experiment focused on two factors: labeling of algorithmic steps with text descriptions and labeling of actions on and changes to the data structure with color and change cues. In this experiment, color was found to be a distractor with regard to labeling of algorithmic actions. Text descriptions added to the animation were found to be helpful in the animations. Thus, the results

indicated that monochrome animations were preferable for the algorithmic actions while text labels increased performance on questions which dealt with conceptual issues. In light of these results, it was decided that future animations would include the textual algorithm labels while avoiding use of color to label data structure changes.

### 10.2.3 Guidelines

The first and second series of design experiments led to several guidelines:

1. Use data representations which are similar to the user's previous experience – for example, vertical bars are parallel in meaning to bar graphs or histograms.

2. Use labels on data items when this is practical. Students prefer this and it eliminates any chance of confusion about the relative value of data elements.

3. Use medium data set sizes, large enough to show several aspects of the animation and small enough not to overload the user's visual processing capacity.

4. Let students have control over animations, particularly the data. One of the advantages of individualized use of computers in instruction is the feeling of control the student has. This previously noted result is mirrored in the effects noted in these experiments. This sense of control may contribute to a higher level of mental involvement and to more memory cues being laid down for later access.

5. Choose animations which have a natural visual mapping of the underlying algorithm to the animation. Results in the experiments were much stronger for the graph related algorithm. This suggests that while animations may be useful for many tasks and domains, they are most useful where the algorithm possesses an natural visual mapping.

6. Be careful in the use of color in the animation to provide cues for action steps of the algorithm. The use of color can be problematic. In this research, color proved to be distracting for learners. Color, while an attention getter, may contribute to information overload.

7. Use monochrome clues such as blinking, shading, or line-style changes (thin, dotted, thick, double) instead of color changes. Such changes help students understand the state of the data structure and the current progress of the algorithm without adding additional distraction.

8. Include a high level text description of algorithmic steps in the animation. These should be brief, active voice and include an indication of the current step. The use of this technique allows visual and textual instructions to complement and reinforce each other.

9. Use vertical bars when representing arrays of numeric data. Vertical bars are very good for giving relative magnitude of data values.

10. Use labels to indicate data values. This is very good for discriminating between data values which are extremely similar in magnitude.

11. Allow students to create example data sets. This provides a feeling of control. In addition, the student can tailor the data set to answer questions as they arise.

### 10.2.4 Use in Teaching

The final segment of experimentation was in the area of how animations are to be used in actually teaching algorithms. One pilot experiment addressed the question of whether text or animation should be presented first. That is, is the animation better used as precursor of the text explanation to come or as a reinforcement of what has already explained? An experiment in which text first was contrasted with animation first indicated that there was no significant difference in performance on the post-test. Due to the pilot nature of this experiment, it was decided to select the text first approach which led to a non-significantly higher average.

A follow-up study dealt with the question of whether animation should be used alone, text alone, or text and animation together. It was interesting to observe in the discovery learning segment of this pilot study that any of the three approaches were sufficient for basic

understanding of simple algorithms such as Radix Sort and Selection Sort. However, more complex algorithms such as Quick Sort could not be understood from animation alone.

These studies were preparatory to a larger experiment using simulated classroom conditions. Animations were designed using the guidelines previously established of textual conceptual steps to accompany monochrome animations. Text was presented first, followed by the animation, as suggested by the pilot studies. A laboratory session was made available to some students in both the active and the passive mode. Results indicated that it was not the extra laboratory session which improved scores on the post-test but rather the interactive experience of self-created example data sets. Those in the interactive laboratory group were found to have higher scores on the post-tests.

## 10.3 Guidelines for Using Algorithm Animations in Teaching

There are also implications for the use of such animations in designing curricula. These animations may best be used as part of a laboratory session concerning the algorithm. With a set of animations available, a student would be able to select the animation he or she needs. Such use of the animations should be in an active lab setting where each student is able to create a data set, enter it, and observe the action of the algorithm on this data.

In summary the design of animations can be guided by the use of both student preferences and performance results. Animations can be more effectively designed if the guidelines given in this research are adapted to the algorithm at hand. Results also indicate that user testing of new animations is advisable to assure that the particular algorithm being used is represented in a manner that is clear to the potential users.

This work indicates that animations may be profitably used in teaching algorithms if they are carefully designed. These animations have value in several areas, especially for algorithms which provide a natural visual mapping.

## 10.4 Implications

With the new curriculum for Computer Science (ACM/IEEE-CS 91) [61] comes a focus on breadth of learning and exposure of the beginning computer science student to many varied concepts and areas of computer science. Animations may well be used to enhance this breadth exposure and to enable the student to grasp an understanding of the field through an understanding of its underlying algorithmic processes.

Additionally, one of the stresses of the new curriculum approaches is more closed laboratory sessions during the course of the computer science major. Animated algorithms could well be employed in these closed laboratories to provide enhancement and reinforcement to lecture and textbook material.

Because of the individual nature of use, these animations might also be made available for discovery or self-directed learning. This would allow the student to use the software when learning a new algorithm or to reinforce some concepts which had been presented in the textbook or in lecture. The flexibility of such a system presents many possibilities.

## 10.5 Future Work

A viable approach to the study of animated algorithms would be to continue to use these animations in teaching computer algorithms. Additional factors to be considered might be coordination of design details to particular algorithms. Another factor that would be of interest is whether these animations should be made available in a structured closed laboratory setting or as an open laboratory option available upon perceived need of the student.

One area of study would be to design a library of animations and make it possible for students to use and modify them on an individual basis. Observational data of how students perceive that these animations could be improved would provide insight into design of new animations.

New animations should be "field-tested" on students to ensure that the student user makes the same or similar mappings to the conceptual base that are intended by the

animation designer.

Since student involvement is a positive enforcement, another area of student involvement would be animation creation. A student XTango shell has been developed by Dr. Stasko which allows students to easily create an animation to explore an algorithm and its workings. Such a student shell would provide another level of involvement for the user. A possible area of study would be how use of such a shell influences understanding of the algorithms animated. An extension to this would be the ability to dynamically alter the animation as it ran, perhaps changing the data representation or which steps of the algorithm are visually illustrated.

Currently a tutorial package is being developed to incorporate these animations into a Data Structures and Algorithm Analysis course. This tutorial will be modified to meet the design and usage guidelines presented here and used in conjunction with such a course.

## 10.6 Conclusion

This line of research clearly demonstrates the value of empirical research in the design of animations. Guidelines for the development and use of algorithm animations were derived. Suggestions for integration of this technology into a learning situation have been justified. Future work in this area holds great promise for the empowerment of the student. Development in virtual reality input and output devices opens further vistas for exploration. It is imperative that we grasp this opportunity to maximize the measurable benefits of the research in order to aid students and generate further interest in the research community.

# APPENDIX A

# CONSENT FORM

Variations of this consent form were used in all experiments described in this research. Any variations were due to changes in the experimental task. All versions informed the subject of his/her rights to confidentiality, the nature of the experimental task, the risk and discomfort level involved, and the benefits of participation.

# CONSENT FORM

I volunteer to serve as a participant in a research experiment on user-computer interaction conducted under the supervision of Dr. A. N. Badre. I realize that no report from this study will contain data that can be identified with me individually; all information relating to individual responses will be treated confidentially. I also realize that the experiment session will have a duration of approximately one hour, and that I can quit the experiment at any time without penalty. At the conclusion of the study I can receive a report of the results if I want them.

During this experiment I will be asked to watch a demonstration of a computer algorithm. At the end of the session, I will fill out questionnaires. The questionnaire will contain questions about the algorithm as well as about my experience with computers and programming.

I understand that the risk and discomfort will not exceed that of normal office work.

If I have any questions, I can call the experimenter or send e-mail, Andrea W. Lawrence, Work 853-9394, Home 758-7709, alawrenc@cc. I understand that I am free to deny any answers to specific items or questions in the questionnaires and other materials.

Upon completion of my participation in this experiment, I will be given course credit as agreed with my professors and department.

_____

Print Name

_____

Signature

_____

Student number

# APPENDIX B

## PREFERENCE SURVEY

The following questions were given to students at the Georgia State University in connection with the first two studies involving student comparison and ranking of varied representations of Quick Sort. The latter questions were also used later in the experiment at the Georgia Institute of Technology which compared student performance on animations of the Quick Sort algorithm which differed in data representation and data set size.

NAME _____

Circle the answer you feel best describes your opinion.

1. How familiar were you with the quick sort algorithm before this lesson?
   Very Familiar  Slightly Familiar  Totally Unfamiliar

2. How well do you feel you now understand the quick sort algorithm presented to you?
   Very Well  Well  OK  Poorly  Very Poorly

3. How well did you understand what each bar or dot represented?
   Very Well  Well  OK  Somewhat Unclear  Completely Unclear

4. Do you feel that the video represented the data well?
   Very Well  Well  Neutral  Poorly  Very Poorly

5. What did you think about the speed of the animation?
   Too Fast  A Little Fast  Just Right  A Little Slow  Much Too Slow

6. In your opinion, how was the length of the animation?
   Much Too Short  A Little Short  Just Right  Too Long  Much Too Long

Select the answer which you feel best completes the sentence.

1. You feel that use of the animation system you just saw
   _____ a) would help you in understanding algorithms.
   _____ b) would hinder you in understanding algorithms.
   _____ c) would make no difference.

2. You would like to see similar demonstrations
   _____ a) frequently during a course.
   _____ b) occasionally during a course.
   _____ c) never during a course.

3. Compared to the way you have been taught algorithms in the past, you think this method to teach this material
   _____ a) much better.
   _____ b) somewhat better.
   _____ c) about the same.
   _____ d) worse.
   _____ e) much worse.

4. How do you think the demonstration could be changed to increase your understanding of the algorithm?

   _____

   _____

5. You have used or seen demonstrated an animation system like the one just presented to you
   _____ a) very often.
   _____ b) often.
   _____ c) on occasion.
   _____ d) a few times.
   _____ e) never used one.

# APPENDIX C

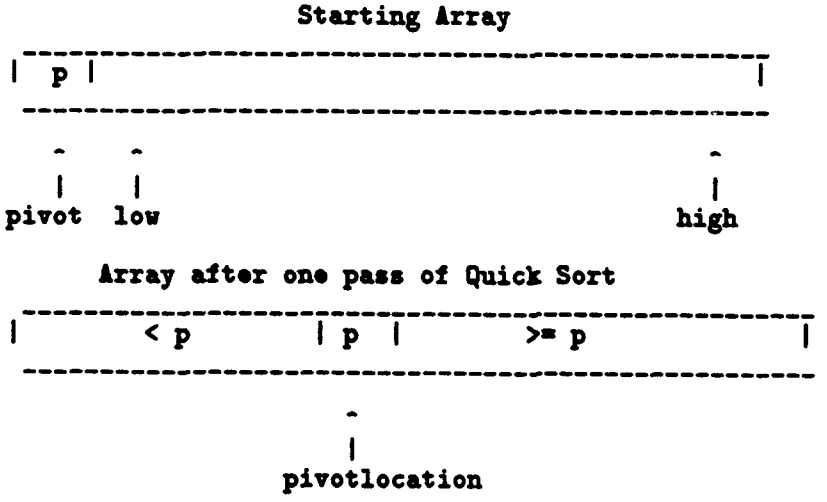
# QUICK SORT DESCRIPTION

The following description of Quick Sort was used in the preference studies at Georgia State and in the first experiment at Georgia Tech which focused on data representation and data set size. Quick Sort was also used in the studies focused on labeling data elements and data set control.

Quick sort performs well in the average case, $O(n \log n)$, although its worst case performance is $O(n^2)$. It is a recursive method which takes advantage of the "divide and conquer" approach, where a structure is divided into two pieces by some criterion, then the two pieces are attacked separately.

Quick sort is a sorting technique in which the list is first partitioned into lower and upper sublists for which all keys are, respectively, less than some pivot key or greater than the pivot key.

Given a pivot value p, we must rearrange the entries of the array and compute the index pivotlocation so that when the pivot is at pivotlocation, all entries to its left have keys less than p, and all entries to the right have larger keys. To allow for the possibility that more than one item has key equal to p, we insist that the items to the left of pivotlocation have keys strictly less than p, and the items to its right have keys greater than or equal to p, as shown in the following diagram:



To reorder the items this way, we must compare each key to the pivot. We start two cursors moving: one will move rightward from the left end of the array, the other leftward from the right end. The rightward-moving cursor (which we'll call 'down' or 'low') will keep moving as long as the elements it scans are smaller than the pivot. The leftward-moving cursor (which we'll call 'up' or 'high') will keep moving as long as the elements it scans are greater than or equal to the pivot.

If the 'up' or 'high' cursor finds a value less than the pivot and the 'down' or 'low' cursor finds one greater than the pivot, those two values are exchanged. Then the cursors are started again from those points.

Eventually, the two cursors will meet. At the point where they meet, all values to the left are guaranteed to be less than the pivot and all values to the right are guaranteed to be greater than the pivot. We call that meeting point the 'partition point' or pivot location. The pivot is swapped with the last smaller element. Then the process is repeated on the left and right portions of the list.

The first item in the list may be chosen to be the pivot, however, any item in the list may be the pivot.

Steps for Quick Sort:

1. Select a pivot element.
2. Start a pointer moving from the left, searching for a value larger than or equal to the pivot.
3. Start a pointer moving from the right, searching for a value smaller than the pivot.
4. When these are located, swap the two elements.
5. Continue moving the pointers from left and right, searching for the larger/equal and smaller elements.
6. Repeat the swap process.
7. The location where the pointers meet is called the partition point. Swap the pivot with the last smaller element.
8. Divide the list into left partition, right partition, and pivot. Repeat the process on each partition piece until each piece contains only one element.
9. Combine the pieces to obtain the sorted list.

# APPENDIX D

# QUICK SORT POST-TEST

The following questionnaire is typical of those used with the Quick Sort algorithm. This algorithm was used in several of the experiments, including Data Representation and Data Set Size, Labeled Data Elements, and Data Set Control.

1. True or False: In each iteration of quick sort, only the pivot is exchanged with other elements.
   _____True
   _____False

2. Which elements are first to be exchanged? (Use 19 as the pivot.) 19 80 2 46 16 12 64 22 17 66 27 35
   _____19 and 80
   _____19 and 17
   _____80 and 17
   _____35 and 80

3. True or False: In Quick Sort, any numbers greater than the pivot will not be moved during one iteration.
   _____True
   _____False

4. The effect of partitioning is to place:
   _____all numbers less than or equal to the pivot in left partition
   _____all numbers in sorted order
   _____half numbers in left partition
   _____all numbers larger than pivot in sorted order

5. True or False: One result of the partitioning process is to place the partition element in its final position.
   _____True
   _____False

6. Given the list 12 5 9 18 2 23 13 10 5 7 31. What pair of numbers would be exchanged second?
   _____18 and 7
   _____23 and 5
   _____12 and 10
   _____none of the above

7. Quick Sort could be most easily programmed using:
   _____selection
   _____recursion
   _____iteration
   _____none of the above

8. Which selection represents the first partitioning by Quick Sort? 8 6 12 20 2 5 47 14
   _____Left 2 6 5 Pivot 8 Right 20 12 47 14
   _____Left 2 5 6 Pivot 8 Right 12 14 20 47
   _____Left 6 2 5 Pivot 8 Right 12 20 47 14
   _____Left 6 5 2 Pivot 8 Right 47 20 14 12

9. The results of the first split of the list are Left 3 1 0 4 5 2 Partition 6 Right 8 9 11 20. Which two numbers will be the second pair exchanged ?
   _____11 and 20
   _____3 and 4
   _____2 and 4
   _____3 and 2

10. Given the list K Q W A B E Y C D U M the results of the first partitioning by Quick Sort are:
    _____E D C A B —— K —— Y W Q U M
    _____B D C A E —— K —— Y Q Q U M
    _____A B C D E —— K —— Q W Y U M
    _____None of the abo·

11. The first partition of the list 4 5 9 3 6 11 7 is 3 —— 4 —— 9 5 6 11 7. Which of these is the next partition?
    _____3 4 6 5 —— 9 —— 7 11
    _____3 4 7 5 6 —— 9 —— 11
    _____3 4 5 6 7 —— 9 —— 11
    _____none of these

12. In Quick Sort, the main advantage is that:
    _____The Divide and Conquer strategy allows work on segments of the data set
    _____Only one number is moved during each iteration
    _____All numbers greater than the pivot are placed in the left-hand partition
    _____All numbers of the data set are compared during each iteration

# APPENDIX E

# PRETEST

The pretest below was given to students participating in the first experiments at Georgia Institute of Technology. The focus of the questions was skills needed to read an algorithmic description and apply the algorithm.

NAME _____

The following describes a procedure:

```
Procedure 1: To change a number:
1.  If the number is zero then the answer is one
2.  Otherwise, the answer is equal to the result of multiplying
the number by the result of applying this procedure to the
number minus one.
```

1. What is the result of change applied to 4?
   _____ a) 64
   _____ b) 256
   _____ c) 1
   _____ d) 24

2. What is the result of change applied to 0?
   _____ a) 1
   _____ b) 0
   _____ c) -1
   _____ d) 64

```
Procedure 2: This method requires a list of numbers and two given numbers
1.  The first given number is equal to one.
2.  Search the list, one element at a time.
3.  Whenever a member of the list is equal to the second given number,
increase the value of the first number given by one.
4.  The result of Procedure 2 is the final value of the first number.
```

3. What is the result of Procedure 2 where the list consists of (12, 3, 2, 6, 3, 14,33) the second number given is 3?
   _____ a) 33
   _____ b) 2
   _____ c) 3
   _____ d) 8

4. The purpose of Procedure 2 is
   _____ a) count the number of elements in the list
   _____ b) return the number of duplicates of the second number
   _____ c) return the number of duplicates of the second number plus one
   _____ d) return the last member of the list

```
Procedure 3:
1.  Take a list with a certain number of members.
2.  For each item of the list,
      if the item is greater than the next item swap the two items.
3.  Continue until the next to last item is compared to the last item.
4.  Repeat this process; however, at each repetition
stop one item closer to the beginning of the list.
5.  The final step of the procedure comes when only two members are
     left in the list.  Compare them; swap if necessary; then end.
```

5. Which members of the list would be first to be exchanged ?

```
19      30     -2      15     108      50    1
```

   _____ a) 19 and 30
   _____ b) 50 and 1
   _____ c) -2 and 15
   _____ d) 30 and -2

# APPENDIX F

# SELECTION SORT

The following description of Selection Sort was used in some early studies at Georgia State as well as in the Labeled Data Experiment of this sequence.

Information is often placed in alphabetical or numerical order to aid a reader in locating a target value. Sorting has been widely studied by computer scientists and many techniques are available. One such technique is called selection sort.

Consider an array with N elements, subscripted 1 .. N.

First find the smallest element in the array, and exchange it with the element in the first position.

Next, exclude the first element from further consideration and repeat the process for elements 2 .. N of the array. Thus the smallest of elements 2 .. N (and the second smallest value in the entire array) will be placed at location 2.

This process continues until there are only two elements left to compared. Then elements N - 1 and N of the array are considered for exchange.

This method is called selection sort because it works by repeatedly "selecting" the smallest remaining element. The selection sort algorithm has an average Big O time of $(N^2)$.

1. Begin with the first element; it is the current smallest one. It is the one to use for comparison.
2. Compare this element with each element in turn.
3. If a smaller element is found, make that the one for comparison.
4. Continue steps 2 and 3 to the end of the list.
5. When the end of the list is reached, swap the current smallest with the first element of the list.
6. Starting from step 2, repeat the process beginning with the second element of the list, comparing it with each later element in turn.
7. Swap the smallest at the end of that set of comparisons with the second element.
8. Repeat with each succeeding element until all elements have been placed and the list is ordered.

# APPENDIX G

## INTERACTION EXPERIMENT POST-TEST

One method of determining understanding of algorithms was to have the subject work through a complete example of the algorithm. The following examples were used in the experiment focused on Data Set Control.

1. Show the steps in quick-sorting the given array:
   (Use 9 as the initial pivot) 9 17 1 21 4 25 16

2. Show the steps in finding the Minimum Spanning Tree of the given graph:

1. Of the two algorithms shown, you preferred:
   _____ Quick Sort _____ Kruskal's Minimum Spanning Tree
2. Which of the two animations did you find clearer in demonstrating the steps of the algorithm?
   _____ Quick Sort _____ Kruskal's MST _____ Both the same
3. What would you suggest as improvements to the animation?
   _____
   _____
   _____
4. What did you like most about the animations?
   _____
   _____
   _____

# APPENDIX H

# TRANSFER TASK

The Knapsack Problem was used as the transfer task where the main task was Kruskal's Minimum Spanning Tree. Both algorithms are based on greedy heuristics.

The Knapsack Problem has as its goal to fill, as nearly as possible, a knapsack which will hold a given weight of objects, with maximum profit. Two pieces of information are given about the objects– weight of each object and profit of each object. Each object may be used only once. A whole object must be used, except for the last object. A fractional portion of the last object may be taken to fill the knapsack. For example, if 2 pounds of space are left in the knapsack, and the best object has weight 10, 2/10ths of this object may be taken as the last step in solving the problem:

1. _____ A knapsack has capacity 20. Objects are X1, X2, X3; profits P1 = 10, P2 = 5, P3 = 3; Weight X1 = 12, X2 = 15, X3 = 14. Which would you place in the knapsack first?

2. _____ How would you select the first object for the knapsack?
   a) the object with the largest weight
   b) the object with the smallest weight
   c) the object with the largest ratio of profit to weight
   d) the object with the largest profit

3. I am filling a knapsack with capacity 30. I have 6 objects. How would you go about selecting which object is first to go into the knapsack?

4. Why would it be difficult to solve this problem by trial and error? (That is, taking each object as first, combined with all other possibilities until the best result is found.)

   _____

5. Show the steps in solving the following knapsack problem. Capacity = 20, n = 3
   (p1,p2,p3) = (25, 24, 15)
   (w1,w2,w3) = (18,15,10)

# APPENDIX I

## SELECTION SORT QUESTIONNAIRE

This Selection Sort test is representative of the test used in the experiments for labeling data elements and in the comparison study of animation, text, and animation plus text.

1. The identifying feature of the Selection Sort is that during each pass, _____ element(s) is (are) put in the proper place.
   _____None
   _____At least one
   _____Two or more

2. How many comparisons are required to Selection Sort a list of length 8?
   _____8
   _____16
   _____28
   _____64

3. Which elements would be first to be exchanged in Selection Sort of the list : Q W E R T Y U I O P?
   _____Y and I
   _____Q and E
   _____Q and I
   _____Y and P

4. The first pass (iteration) of a Selection Sort requires at most _____exchanges (where n is the number of elements to be sorted).
   _____n
   _____n - 1
   _____1
   _____n / 2

5. Which represents the result of two iterations of Selection Sort on the given list?
   S O R T E X A M P L
   _____A O R T E X S M P L
   _____A E R T O X S M P L
   _____X T R O E S A M P L
   _____none of the above

6. Given the list 26 24 3 17 25 13 60 47 1, which represents the list after four iterations of Selection Sort?
   _____1 3 13 17 26 24 25 47 60
   _____3 17 24 25 26 13 60 47 1
   _____1 3 13 17 25 24 60 47 26
   _____none of the above

7. Which numbers will be the third pair of numbers compared in the first iteration of Selection Sort given the list 3 4 2 1 7 6 8 5?
   _____3 and 1
   _____2 and 1
   _____7 and 6
   _____none of the above

8. True or False: In Selection Sort, an exchange must be made during each iteration
   _____True
   _____False

9. Consider an already sorted list and an unsorted list, Selection Sort requires how many comparisons?
   _____More for the sorted list
   _____More for the unsorted list
   _____The same for both lists

10. The initial list is 13 12 15 6 10 18 2 4 11 1. Later it is 1 2 4 6 10 18 12 15 11 13. How many iterations have been done?
    _____3
    _____4
    _____either 3 or 4
    _____5

11. Given the list X Q R A B C D E T, which pair of letters would be the second to be exchanged?
    _____X and A
    _____Q and B
    _____B and C
    _____none of the above

12. Which list represents the numbers after 3 iterations of Selection Sort:
    12 9 10 8 2 4 16 7 18?
    _____2 9 10 8 12 4 16 7 18
    _____2 4 7 8 9 10 12 16 18
    _____2 4 10 8 12 9 16 7 18
    _____2 4 7 8 12 9 16 10 18

# APPENDIX J

# RADIX SORT QUESTIONNAIRE

**This questionnaire was used in the pilot study which contrasted text only, animation only, and animation plus text. Questions were based on the radix sort.**

1. How many sublists are needed for radix sort?
   _____no certain number
   _____9
   _____10
   _____5

2. The first time through the list, 342 goes into what sublist?
   _____3
   _____4
   _____2
   _____0

3. What order are the sublists recombined in?
   _____longest first
   _____numerical order
   _____shortest first
   _____no special order

4. The third time through the master list, 19 goes into sublist
   _____1
   _____9
   _____0
   _____3

5. Given the numbers 33 5 12 and 15, what will the master list look like after the first time the list is recombined?
   _____5 12 15 33
   _____33 12 15 5
   _____12 33 5 15
   _____12 5 15 33

6. How many times is it necessary to go through the master list for the following list of numbers? 123 19 9 2 8 101
   _____1
   _____2
   _____6
   _____3

7. What do you do on the second pass though the master list with a single digit number such as 3?
   _____put it in the sublist 3
   _____put it in the sublist 0
   _____do not remove it from the master list
   _____none of the above

8. If the numbers are already sorted before the process begins, will they still be sorted after the first time through the master list?
   _____Yes
   _____No

9. Given the following list, how will it appear after the master list is recombined the first time? 12 4 5 14 144 9 22 39 29 7
   _____12 22 4 14 144 5 7 9 39 29
   _____4 5 7 9 12 14 22 29 39 144
   _____4 5 7 12 14 22 9 39 29 144
   _____12 22 29 39 4 14 144 5 7 9

10. During the third time through the master list, what will be in sublist 3? The original list is : 123 333 12 133 42 15 9 88 103
    _____103, 123, 133, 333
    _____333
    _____123, 103
    _____133, 333

# APPENDIX K

## KRUSKAL FIXED RESPONSE QUESTIONNAIRE PRESENTATION STYLE EXPERIMENT

These questions made up the fixed-response on-line questionnaire. This questionnaire, or portions of it, was used for the Presentation Style Experiment as well as the Active/Passive Design Issue Experiment and the Text First or Algorithm First Experiment.

1. Edges AB, BC, and CA in figure 2 form a cycle
   _____True
   _____False

2. In Kruskal Algorithm, the first step in finding the Minimum Spanning Tree is:
   _____Sort the edges by weight
   _____Select the two shortest edges
   _____Select the shortest edge from node 1
   _____None of the above

3. Consult the figure sheet. In graph 1, what edge will be first to be added to the shortest path?
   _____HG
   _____AB
   _____either HG or AB
   _____neither HG nor AB

4. Consult the figure sheet. In graph 1, which edge will be added to the Minimum spanning Tree second?
   _____GF
   _____HG
   _____CI
   _____none of the above

5. Consult the figure sheet. In graph 1, if edges HG, IC, GF, CF, and AB are already in the path, which edge will be added next?
   _____IG
   _____CD
   _____HI
   _____ None of the above

6. Consult the figure sheet. In graph 2, which edge will be placed in the Minimum Spanning Tree first?
_____DF
_____AC
_____AB
_____None of the above

7. Consult the figure sheet. In graph 2, if edges AC, CF, and DF are already in the path, which edge will be next?
_____BC
_____AD
_____CD
_____none of the above

8. The time complexity of Kruskal's Algorithm seems to be: (let V = number of vertices, E = number of edges)
_____$O(V + E)$
_____$O(E \log E)$
_____$O(E * V)$
_____none of the above

9. In Kruskal's Algorithm the first edge selected for the minimum spanning tree is
_____Connected to the first vertex
_____The edge of least weight
_____The edge of least weight leading from the first vertex
_____none of the above

10. Kruskal's Algorithm selects at each step:
_____The edge of least weight
_____The edge of least weight which neighbors the last edge selected
_____The edge of least weight which does not make a cycle _____None of the above

11. Consult the figure sheet. On graph 3, the minimum spanning tree begins with:
_____ED
_____GI
_____AB
_____either ED or GI

12. Consult the figure sheet. On graph 3, ED, GI, AB, DC, HC, and EI have been added to the Minimum Spanning Tree. Which edge will be added next?
_____GH
_____BC
_____IH
_____none of the above

13. Consult the figure sheet. On graph 4, the correct Minimum Spanning Tree contains:
_____v1-v7, v2-v7, v3-v4, v4-v7, v7-v3, v4-v5
_____v1-v7, v3-v4, v7-v2, v7-v3, v4-v5, v6-v1
_____v1-v7, v7-v2, v7-v4, v7-v5, v7-v6, v7-v3
_____None of the above

14. Kruskal's Algorithm is advantageous over other Minimum Spanning Tree Algorithms because
_____It always finds a least Spanning Tree.
_____It is faster than any other method.
_____It is more efficient in use of data structures.
_____None of the above

15. Consult the figure sheet. On graph 5, if V2-V3, V2-V4 are already in the minimum Spanning Tree, the next edge to add is:
_____V3-V4
_____V2-V6
_____V4-V6
_____None of the above

16. Consult the figure sheet. On graph 5, the correct Minimum Spanning Tree contains:
_____v2-v3, v2-v4, v2-v6, v2-v1, v4-v5
_____v1-v2, v2-v3, v2-v4, v2-v6, v4-v5
_____v2-v3, v2-v4, v3-v4, v2-v6, v4-v5
_____None of the above

17. Consult the figure sheet. On graph 2, the correct Minimum Spanning Tree contains:
_____AC,DF,DA, CB, FE
_____AC,DF,CF,CB,EF
_____AC,FD,CB,AB,CE
_____None of the above

18. Kruskal's Algorithm is the only possible way to find a Minimum Spanning Tree.
_____True
_____False

19. A minimum Spanning tree in a connected graph with 10 nodes has
___ edges.
_____20
_____100
_____9
_____10

20. How familiar were you with the Kruskal algorithm before this lesson?
_____Very Familiar
_____Slightly Familiar
_____Totally Unfamiliar

21. How well do you feel you now understand the Kruskal algorithm presented to you?
_____Very Well
_____Well
_____Poorly
_____Very Poorly

22. You feel that use of the animation system you just saw
_____would help you in understanding algorithms
_____would hinder you in understanding algorithms
_____would make no difference

23. You would like to see similar demonstrations
_____frequently during a course
_____occasionally during a course
_____never during a course

24. Compared to the way you have been taught algorithms in the past, you think this method to teach this material:
_____better
_____about the same
_____worse

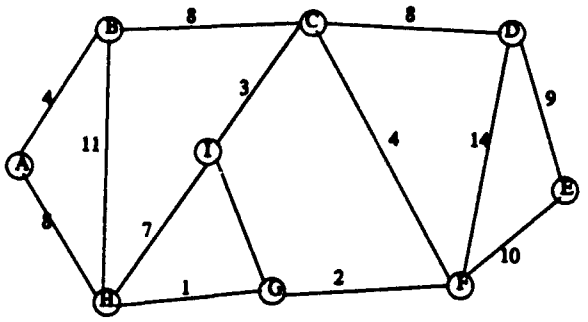



Figure K.1: Graph 1

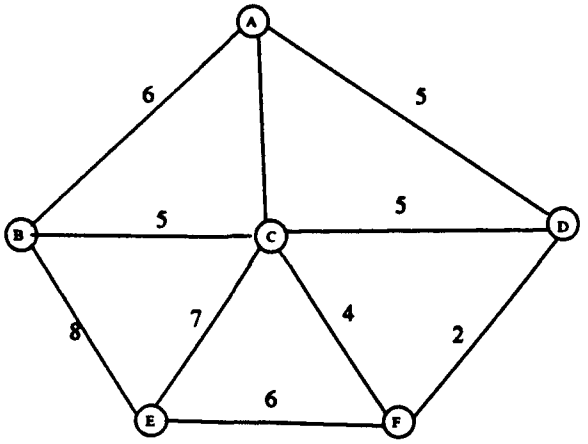



Figure K.2: Graph 2

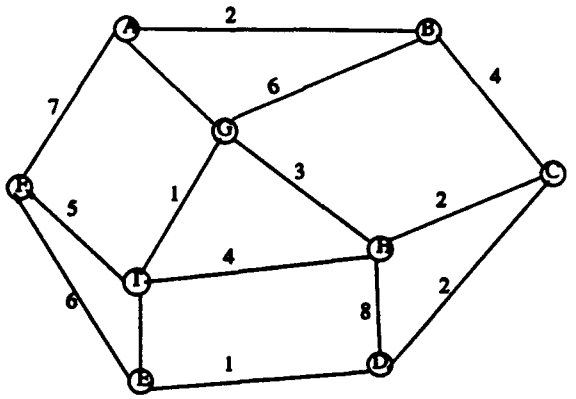



Figure K.3: Graph 3

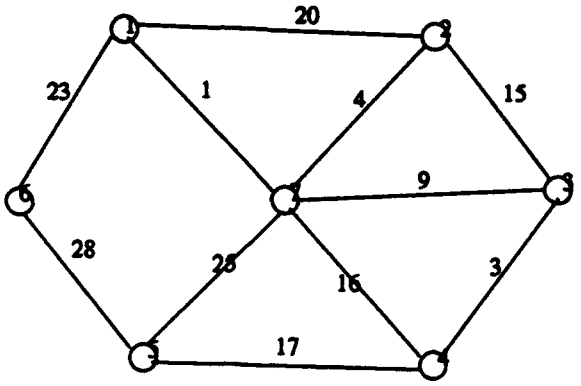



Figure K.4: Graph 4

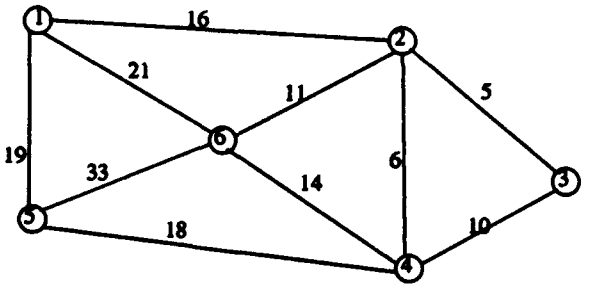



Figure K.5: Graph 5

# APPENDIX L

## KRUSKAL FREE-RESPONSE QUESTIONNAIRE PRESENTATION STYLE EXPERIMENT

The following questions were used as the free-response post-test for Kruskal's Minimum Spanning Tree Algorithm. These questions were given following the computerized fixed-choice segment of the post-test and were a paper handout rather than a computerized sequence. These questions were used for several experiments including the experiment in presentation style where the variables studied were type of example accompanying the lecture and the type of laboratory session (no laboratory, passive laboratory, or active laboratory).

Name _____

1. When would it be possible for either of two edges to be selected as the next edge of the spanning tree?
2. List a combination of edges in the graph which form a cycle.
3. In the graph, edges 1, 2, and 3 have already been added to the MST tree. The next shortest edge is edge 4. Can it be added? Why or why not?
4. Under what conditions would the next shortest edge not be added to the Minimum Spanning Tree?
5. Describe, in your own words, the steps of the Kruskal Minimum Spanning Tree Algorithm.
6. What, in your opinion, is the key part of the algorithm which guarantees the Spanning Tree obtained will be minimal?
7. Apply Kruskal's MST Algorithm to the figure below:

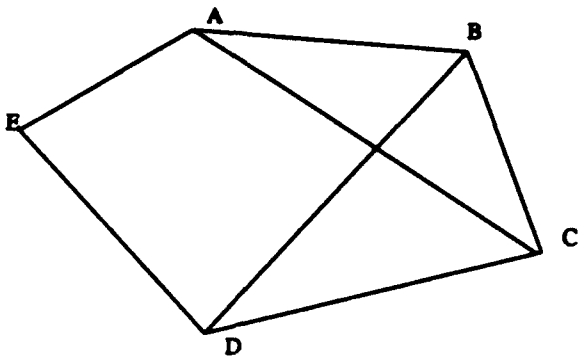



Figure L.1: Graph, Question 2

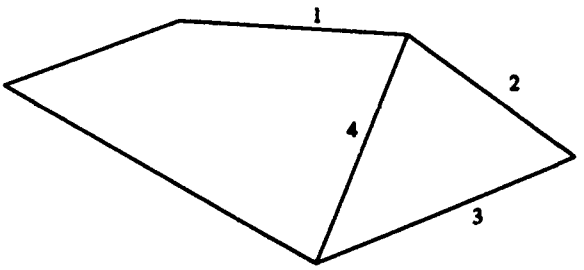



Figure L.2: Graph, Question 3

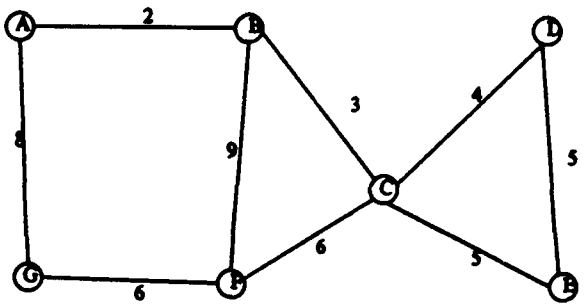



Figure L.3: Graph, Question 7

# BIBLIOGRAPHY


[1] Jeanette Allen. *Effects of Representation on Programming Behavior.* PhD thesis, Georgia Institute of Technology, 1993.

[2] Anthony Anderson and Stephen Draper. An introduction to measuring and understanding the learning process. *Computers in Education*, 17(1):1–11, 1991.

[3] Albert Badre, Margaret Beranek, J. Morgan Morris, and John Stasko. Assessing program visualization systems as instructional aids. Graphics Visualization and Usability Center GIT-GVU-91-23, Georgia Institute of Technology, Atlanta, GA, October 1991.

[4] Albert Badre, Margaret Beranek, J. Morgan Morris, and John Stasko. Assessing program visualization systems as instructional aids. In Ivan Tomek, editor, *Computer Assisted Learning, ICCAL '92*, volume 602 of *Lecture Notes in Computer Science*, pages 87–99, Wolfville, Nova Scotia, Canada, June 1992. Springer-Verlag.

[5] R. M. Baecker and D. Sherman. Sorting out sorting. Film ACM SIGGRAPH, 1981.

[6] Ronald Baecker and Aaron Marcus. Design principles for the enhanced presentation of computer program source text. In *CHI'86 Proceedings*, pages 51–58, New York, April 1986. Association for Computing Machinery, ACM.

[7] Patricia Baggett. Understanding visual and verbal messages. In Mandl and Levin, editors, *Knowledge Acquisition from Text and Pictures*, pages 101–124. Elsevier Science, Amsterdam, 1989.

[8] David Baskerville. Graphic presentation of data structures in the DBX debugger. Technical Report UCB/CSD 86/260, University of California at Berkeley, Berkeley, CA, October 1985.

[9] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[10] H. Bocker, G. Fischer, and H. Nieper. The enhancement of understanding through visual representations. In *CHI '86 Proceedings*, pages 44–50. SIGCHI, 1986.

[11] F. J. Brandenburg, P. Gorny, and M.J. Tauber. Nice drawings of graphs are computationally hard. In *Visualization in Human-Computer Interation: 7th Interdisciplinary Workshop on Informatics and Psychology*, pages 1–15, Scharding, Austria, May 1990.

[12] Marc Brown. Exploring algorithms using Balsa-II. *Computer*, 21(5):14–36, 1988.

[13] Marc H. Brown. Zeus: A system for algorithm animation and multi-view editing. In *Proc. 1991 IEEE Workshop on Visual Languages*. IEEE, October 1991.

[14] Marc H. Brown. An introduction to Zeus: Audiovisualization of some elementary sequential and parallel sorting algorithms. In *CHI '92 Conference Proceedings*, pages 663–665. CHI, May 1992.

[15] Marc H. Brown and John Hershberger. Color and sound in algorithm animation. *Computer*, 25(12):52–63, December 1992.

[16] Marc H. Brown and Robert Sedgewick. Techniques for algorithm animation. *IEEE Software*, 2(1):28–39, January 1985.

[17] Marc H. Brown and Robert Sedgwick. A system for algorithm animation. *Computer Graphics*, 18(3):177–186, 1984.

[18] Richard Catrambone and Keith J. Holyoak. Overcoming contextual limitations on problem solving transfer. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 15(6):1147–1156, 1989.

[19] William Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, September 1984.

[20] Nancy Cuniff and Robert Taylor. Graphical vs textual representation: An empirical study of novices' program comprehension. In G. M. Olson, S. Sheppard, and E. Soloway, editors, *Empirical Studies of Programmers: Second Workshop*, pages 114–131. 1987.

[21] P Duchastel. Illustrating instructional texts. *Educational Technology*, 18:36–39, 1978.

[22] S. K. Feiner and K. R. McKeown. Coordinating text and graphics in explanation generation. In *Proc. AAAI-90*, pages 442–449. AAAI, July 1990.

[23] Richard M. Felder and Linda K. Silverman. Learning and teaching styles in engineering education. *Engineering Education*, 78(7):674–681, 1988.

[24] Mary L. Gick and Keith J. Holyoak. Analogical problem solving. In Gentner and Stevens, editors, *Mental Models*, pages 279–306. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

[25] Sam Glucksberg and Robert W. Weisbert. Verbal behavior and problem solving: Some effects of labeling in a functional fixedness problem. *Journal of Experimental Psychology*, 71(5):659–664, 1966.

[26] Frank Harary. *Graph Theory*. Addison-Wesley, Reading, Massachusetts, 1972.

[27] Esa Helttula, Alulikki Hyrskykari, and Kari-Jouko Raiha. Graphical specification of algortihm animations with ALADDIN. In *22nd International Conference on Sytems Sciences*, pages 892–901. Systems Sciences, 1988.

[28] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Data Structures in Pascal*. Computer Science Press, Rockville, MD, 1984.

[29] Tomihisa Kamada and Satoru Kawai. A general framework for visualizing abstract objects and relations. *ACM Transactions on Graphics*, 10(1):1–39, January 1991.

[30] Donald E. Knuth. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Addison-Wesley, Reading, Massachusetts, 1973.

[31] Jill Larkin and Herbert Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[32] Kenneth Lodding. Iconic interfacing. *IEEE Computer Graphics and Applications*, pages 11–20, March/April 1983.

[33] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, April 1987.

[34] M. Maguire. A review of human factors guidelines and techniques for the design of graphical human-computer interfaces. *Computers and Graphics*, 9(3):221–235, 1985.

[35] A. Mahling, J. Herczeg, H. Bocker, P. Gorny, and M.J. Tauber. Beyond visualization: Knowing and understanding. In *Visualization in HCI*, pages 16–26, Sharding, Austria, 1988.

[36] Richard Mayer. The psychology of how novices learn computer programming. *Computing Surveys*, 13(1):121–141, 1981.

[37] Richard Mayer. Models for understanding. *Review of Educational Research*, 59(1):43–64, 1989.

[38] Brad Myers. Visual programming, programming by example, and program visualization: A taxonomy. In *CHI '86 Proceedings*, pages 59–66, 1986.

[39] Brad Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing*, 1(1):97–124, 1990.

[40] Brad A. Myers. A system for displaying data structures. *Computer Graphics: SIGGRAPH '83*, 17(3):115–125, July 1983.

[41] Susan Palmiter. The effectiveness of animated demonstrations for computer-based tasks: a summary, model and future research. *Journal of Visual Languages and Computing*, 4(1):71–89, 1993.

[42] Susan Palmiter and Jay Elkerton. An evaluation of animated demonstrations for learning computer-based tasks. In *CHI '91 Conference Proceedings*, pages 257–263. CHI, April 1991.

[43] Jenny Preece. Some hci issues concerned with displaying quantitative information graphically. In *Visualization in Human-Computer Interaction 7th Interdisciplinary Workshop on Informatics and Psychology*, pages 209–226, Scharding, Austria, May 1988.

[44] Gerald K. Rambally and Rodney S. Rambally. Human factors in CAI design. *Computers in Education*, 11(2):149–153, 1987.

[45] S. K. Reed. Effects of computer graphics on improving estimates to algebra word problems. *Journal of Educational Psychology*, 77:285–298, 1985.

[46] Lloyd Rieber. Animation, incidental learning, and continuing motivation. *Journal of Educational Psychology*, 83(3):318–328, 1991.

[47] Lloyd Rieber, Mary Boyce, and Chahriar Assad. The effects of computer animation on adult learning and retrieval tasks. *Journal of Computer Based Instruction*, 17(2):46–52, Spring 1990.

[48] Lloyd P. Rieber. Animation in computer-based instruction. *Educational Technology Research and Development*, 38(1):77–86, Spring 1990.

[49] Lloyd P. Rieber. Using computer animated graphics in science instruction with children. *Journal of Educational Psychology*, 82(1):135–140, 1990.

[50] Philip Robertson. A methodology for choosing data representations. *IEEE Computer Graphics and Applications*, pages 56–66, May 1991.

[51] Brian H. Ross. This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology, Learning, Memory, and Cognition*, 13(4):629–639, 1987.

[52] Brian H. Ross. Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology, Learning, Memory, and Cognition*, 15(3):456–468, 1989.

[53] Robert Sedgewick. *Algorithms*. Addison-Wesley, Reading, Massachusetts, 1988.

[54] John Sparrow. Graphical displays in information systems: Some data properties influencing the effectiveness of alternative forms. *Behaviour and Information Technology*, 8(1):43–56, 1989.

[55] John Stasko. XTango, 1.42, C source code with example scripts running on Unix workstations running the X Window System. Available by anonymous ftp from par.cc.gatech.edu in /pub.

[56] John Stasko. TANGO: A framework and system for algorithm animation. *Computer*, 23(9):39–44, 1990.

[57] John Stasko, Albert Badre, and Clayton Lewis. Do algorithm animations assist learning? An empirical study and analysis. In *INTERCHI '93 Conference Proceedings*, pages 61–66. CHI, April 1993.

[58] John T. Stasko. The Path-Transition Paradigm: A practical methodology for adding animation to program interfaces. *Journal of Visual Languages and Computing*, 1(3):213–236, September 1990.

[59] John T. Stasko. Animating algorithms with XTANGO. *SIGACT News*, 23(2):67–71, Spring 1992.

[60] John T. Stasko and Charles Patterson. Understanding and characterizing software visualization systems. In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pages 3–10, Seattle, WA, September 1992.

[61] A. Tucker, B. Barnes, R. Aiken, K. Barker, J. Cain, S. Conry, G. Engel, R. Epstein, D. Lidtke, M. Mulder, J. Rogers, E. Spafford, and A. Turner. *Computing Curricula 1991*. ACM/IEEE-CS Joint Curriculum Task Force, ACM and IEEE-CS Press, New York, 1991.

[62] Edward Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Box 430, Cheshire, Connecticut 06410, 1983.

[63] Philip Vanneste and Henk Olivie. Towards an intelligent environment to learn programming (in pascal). in press, 1989.

[64] Bernd Weidenmann. When good pictures fail: An information-processing approach to the effect of illustrations. In Mandl and Levin, editors, *Knowledge Acquisition from Text and Pictures*, pages 157–170. Elsevier Science, Amsterdam, 1989.

[65] Roger E. Whitney and N. Scott Urquhart. Microcomputers in the mathematical sciences: Effects on courses, students, and instructors. *Academic Computing*, 4(6):14–18, March 1990.

[66] Christopher Wickens and Anthony D. Andre. Proximity compatibility and information display: Effects of color, space, and objectness on information integration. *Human Factors*, pages 61–77, February 1990.

[67] W. Winn. The effect of block-word diagrams on the structuring of science concepts as a function of general ability. *Journal of Research in Science Teaching*, pages 201–211, 1980.

[68] W. Winn. The design and use of instructional graphics. In Mandl and Levin, editors, *Knowledge Acquisition from Text and Pictures*, pages 125–144. Elsevier Science, Amsterdam, 1989.

# VITA

Andrea W. Lawrence

B. S., Mathematics: Purdue University

M. S., Computer Science: Atlanta University

Ph.D., Computer Science, Georgia Institute of Technology

Andrea Williams Lawrence is a native of Asheville, North Carolina. She was born on October 6, 1946. She attended Spelman College, where she majored in mathematics. Ms. Lawrence holds the B.S. in Mathematics from Purdue University (1970) and the M.S. in Computer Science from Atlanta University (1985). In addition, she has completed graduate studies in Education at the University of Cincinnati and short courses in image processing, artificial intelligence, parallel processing, and graphics.

She has taught in the Cincinnati Public Schools and is currently a faculty member in the Computer Science Department at Spelman College in Atlanta, Georgia. Andrea Lawrence has taught both Mathematics and Computer Science at Spelman. She has served as Director of Computer Literacy and Director of the Computer and Information Sciences Program.

Andrea Lawrence completed studies for the Doctor of Philosophy in Computer Science at the Georgia Institute of Technology in September, 1993. Dr. Albert N. Badre served as chairman of her dissertation committee. Her current research interests are in the area of Human Computer Interaction. One focus of her work has been the use of visualization and computer animation in teaching computer science.

# VITA

Andrea W. Lawrence

B. S., Mathematics: Purdue University

M. S., Computer Science: Atlanta University

Ph.D., Computer Science, Georgia Institute of Technology

Andrea Williams Lawrence is a native of Asheville, North Carolina. She was born on October 6, 1946. She attended Spelman College, where she majored in mathematics. Ms. Lawrence holds the B.S. in Mathematics from Purdue University (1970) and the M.S. in Computer Science from Atlanta University (1985). In addition, she has completed graduate studies in Education at the University of Cincinnati and short courses in image processing, artificial intelligence, parallel processing, and graphics.

She has taught in the Cincinnati Public Schools and is currently a faculty member in the Computer Science Department at Spelman College in Atlanta, Georgia. Andrea Lawrence has taught both Mathematics and Computer Science at Spelman. She has served as Director of Computer Literacy and Director of the Computer and Information Sciences Program.

Andrea Lawrence completed studies for the Doctor of Philosophy in Computer Science at the Georgia Institute of Technology in September, 1993. Dr. Albert N. Badre served as chairman of her dissertation committee. Her current research interests are in the area of Human Computer Interaction. One focus of her work has been the use of visualization and computer animation in teaching computer science.